宇宙X線観測用放射線検出器
多チャンネル読み出しシステムの開発

広島大学 理学部 物理科学科
1479043A
田中琢也
高エネルギー宇宙・素粒子実験研究室
主査： 深澤泰司　副査： 岡本宏己

2006 年 2 月 10 日

概 要

超新星、ブラックホール、銀河・銀河団などの天体からは、粒子加速による硬 X 線から軟ガンマ線のエネルギー領域での非熱的な放射が観測されている。粒子の加速機構の解明は、現代の宇宙物理学における重要なテーマの一つである。このエネルギー領域での感度の良い観測を行なうため、コンプトンカメラと呼ばれる観測装置が考案され、すざく衛星に続く次期 X 線天文衛星である NeXT への搭載を目指して開発が進められている。コンプトンカメラには両面シリコンストリップ検出器 (DSSD) が用いられるが、DSSD からの出力は合わせて数千チャンネルにも及ぶため、衛星に搭載することのできる小型で高性能な読み出しシステムの開発が必要である。そのため、DSSD と同時に新しい読み出しシステムの開発も進められている。また、DSSD から読み出されたデータは衛星上のデータ処理装置などに転送する必要がある。現在、衛星上での機器間統一通信規格である SpaceWire の開発が世界的に進められており、NeXT でも採用が検討されている。

本研究では、新しい読み出しシステムの立ち上げをスムーズに行なうため、DSSD の読み出しに関する基礎を調べた。また、SpaceWire の開発プラットフォームである SpaceCube と、SpaceWire を搭載した ADC ボードを入手し、ADC ボード上の FPGA にハードウェア記述言語 VHDL を用いてマルチチャンネルアナライザの機能を実装することにより、SpaceWire で制御できるマルチチャンネルアナライザを開発した。そして、将来 SpaceWire で DSSD の読み出し回路を制御するための第一歩を踏んだ。

目 次

第 1 章　序論 **5**
1.1　背景 5
1.2　研究の目的 6

第 2 章　DSSD **7**
2.1　半導体検出器 7
2.2　シリコンストリップ検出器 8
2.2.1　シリコンストリップ検出器の構造 8
2.2.2　リーク電流 9
2.2.3　エネルギー分解能 9
2.3　半導体多層コンプトンカメラと NeXT 衛星 12
2.3.1　コンプトンカメラ 12
2.3.2　NeXT 衛星 14

第 3 章　VA32TA と VA-DAQ を用いた DSSD 多チャンネル読み出しシステム **15**
3.1　VA32TA 15
3.1.1　VA32TA の特徴 15
3.1.2　VA32TA の読み出しシーケンス 16
3.2　VA-DAQ 17
3.2.1　VA-DAQ の機能 18
3.2.2　VA-DAQ を用いた多チャンネル読み出し 18
3.3　読み出しの実際 19
3.3.1　4 cm DSSD でのセットアップ 19
3.3.2　解析方法 20
3.3.3　測定結果の例 21
3.4　VA-DAQ の問題点 23
3.5　新型 DSSD 読み出しシステム 24
3.5.1　フローティング 24
3.5.2　新型読み出しシステムの特徴 25

第 4 章　SpaceWire **27**
4.1　SpaceWire の特徴 27

4.2 SpaceCube と FADC ボード 29
4.2.1 SpaceCube 29
4.2.2 FADC ボード 30
4.2.3 FPGA 31
4.2.4 VHDL 33
4.3 SpaceCube と FADC ボードを用いたマルチチャンネルアナライザの製作 35
4.3.1 SpaceCube の立ち上げ 35
4.3.2 MCA の開発 (ハードウェア) 36
4.3.3 MCA の開発 (ソフトウェア) 47
4.4 製作した MCA の評価 48
4.4.1 リニアリティ 48
4.4.2 デッドタイム 49
4.4.3 トリガの判定 51
4.4.4 スピード試験 51
4.4.5 ガンマ線源の測定 54

第 5 章 まとめと今後 60

図 目 次

2.1 半導体検出器の原理 8
2.2 基本的な両面シリコンストリップ検出器の構造 [11] 8
2.3 DC 結合の等価回路 10
2.4 AC 結合の等価回路 10
2.5 DSSD が持つ容量 [11] 11
2.6 吸収係数 [14] 12
2.7 コンプトンカメラ [11] 13
2.8 NeXT 衛星の予想図 [14] 14
2.9 SGD の予想図 [14] 14

3.1 VA32TA チップ [11] 15
3.2 VA32TA のブロック図 [11] 16
3.3 VA32TA の読み出しタイミングチャート 17
3.4 VA-DAQ [11] 17
3.5 VA32TA と VA-DAQ の接続 19
3.6 4 cm DSSD 20
3.7 セットアップ 20
3.8 p 側ストリップのペデスタル・コモンモードノイズ補正前のスペクトル 22
3.9 p 側ストリップのペデスタル・コモンモードノイズ補正後のスペクトル 22
3.10 n 側ストリップのスペクトル (補正後) 22
3.11 p 側ストリップのエネルギー分解能 22
3.12 線源のイメージ 23
3.13 フローティング 25
3.14 新しい読み出しシステムのブロック図 26

4.1 Data-Strobe(DS) encoding 28
4.2 ケーブルの構造 [5] 28
4.3 コネクタ・ピンアサイン [5] 28
4.4 SpaceWire のネットワーク 29
4.5 SpaceCube の写真 30
4.6 FADC ボードの写真 31
4.7 FADC ボードのブロック図 31

4.8 ロジック・アレイのブロック図 [1] 32
4.9 ロジック・エレメントのブロック図 [1] 33
4.10 VHDL でロジック・ゲートを記述した例 35
4.11 セットアップ全体のブロック図 36
4.12 ピークホールド信号生成のタイミングチャート 37
4.13 NIM モジュール 38
4.14 FADC ボードと SpaceCube 38
4.15 セットアップの全体 38
4.16 MCA のブロック図 39
4.17 トリガとピークホールド出力のタイミング 40
4.18 詳細なブロック図 42
4.19 データ取得の状態遷移図 42
4.20 生成した FIFO の動作 45
4.21 バッファ読み出しの状態遷移図 46
4.22 読み出しレジスタのデータフォーマット 46
4.23 データ収集中の PC 画面 48
4.24 ch1 のリニアリティ 49
4.25 ch2 のリニアリティ 49
4.26 1 イベントごとのデッドタイム 50
4.27 バッファが溢れる直前の busy 信号 51
4.28 バッファが溢れたときの busy 信号 51
4.29 イベントレート 10 kHz でのスペクトル 53
4.30 イベントレート 30 kHz でのスペクトル 53
4.31 同期していない信号 (2.0 kHz) を入れたときの ch1 のスペクトル 54
4.32 同期していない信号 (1.0 kHz) を入れたときの ch2 のスペクトル 54
4.33 同期していない信号 (7.0 kHz) を入れたときの ch1 のスペクトル 54
4.34 同期していない信号 (17 kHz) を入れたときの ch2 のスペクトル 54
4.35 CsI 検出器 55
4.36 ^{137}Cs のスペクトル 56
4.37 ^{22}Na のスペクトル (下は coincidence を取ったもの) 56
4.38 コンプトン散乱測定のセットアップ 57
4.39 θ が 90 度のときのスペクトル 58
4.40 θ が 45 度のときのスペクトル 58
4.41 θ が 90 度のときの 2 次元ヒストグラム 59
4.42 θ が 45 度のときの 2 次元ヒストグラム 59

第1章　序論

1.1　背景

超新星残骸やブラックホール、銀河団などの天体からは、硬X線・ガンマ線の放射が観測されている。これまでの観測の結果、この放射は相対論的エネルギーまで加速された粒子による制動放射、シンクロトロン放射、逆コンプトン散乱など、非熱的な過程によるものであることが分かってきた。非熱的な放射は宇宙のあらゆる場所で確認されており、粒子の加速機構などの解明は、現代の宇宙物理学上たいへん重要な問題である。

現在、数10 keV以下のエネルギー領域では、集光・結像のできるX線望遠鏡を使った観測が行われている。しかし、数10 keV以下の領域では熱的な放射が大部分を占め、非熱的な放射はあまり観測されない。また、非熱的な放射が主体となる数10 keVから数10 MeVまでの硬X線から軟ガンマ線までの領域では、光子と物質との相互作用が起きにくく、相互作用もコンプトン散乱が支配的である。そのため、このエネルギー領域では光子の持っていたエネルギーや到来方向の精度の良い測定が困難であった。そこで、コンプトン散乱を積極的に利用したコンプトンカメラと呼ばれる装置が新たに考案された。これは、検出器内で起こったコンプトン散乱を再構成することにより、検出器に入射した光子の到来方向とエネルギーを正確に決定するという装置である。

現在、すざく衛星に続く次期X線天文衛星NeXT計画が、宇宙航空研究開発機構・宇宙科学研究本部(JAXA/ISAS)を中心に進められている。NeXT衛星には両面シリコンストリップ検出器(DSSD)を用いた半導体多層コンプトンカメラの搭載が検討されており、我々のグループも開発に参加している。DSSDは一枚あたり、両面合わせて数10から数100チャンネルの信号出力を持ち、ガンマ線の入射位置とエネルギーを同時に測定することができる検出器である。半導体多層コンプトンカメラにDSSDを用いる場合、DSSDを数10枚重ねて使用することになるので、数1000にも及ぶチャンネルからの信号を高速に読み出す必要がある。また、読み出し後の信号処理はコンプトンカメラの測定精度にも影響をおよぼすため、DSSDそのものに加えて読み出しシステムの性能も重要である。さらに衛星への搭載を考えた場合、読み出しシステムには低ノイズ、小型、軽量、低消費電力、安定性、他の機器との柔軟な接続といったものも要求される。そのため、現在DSSDと合わせて衛星搭載を考慮した読み出しシステムの開発も進められている。

半導体多層コンプトンカメラでは、非常に多くのチャンネルからの信号を高速に読み出すため、単位時間あたりに得られるデータも膨大な物となる。そのため、得られたデータを衛星上の他の装置に転送するには高速な通信回線が必要となる。従来の衛星では、衛星ごと、搭載装置ごとに異なった通信規格が混在していたため、開発期間の長期化、コストの増大、信頼性の低下などといった問題が生じていた。このような問題を打開するため、衛星上での装置間通信規格の統一が考えられ、現在SpaceWireと呼ばれる統一規格の策定が世界的に進められている。SpaceWireは

高速な通信を行える一方で、センサ等の小型の装置から観測装置、姿勢制御装置などあらゆる装置間の通信に対応できる柔軟性を持っている。日本からも JAXA/ISAS を中心に SpaceWire の規格策定に参加しており、NeXT 衛星では SpaceWire の採用が検討されている。そのため、NeXT 衛星に搭載される観測装置には SpaceWire への対応が求められる。

1.2 研究の目的

衛星に搭載される半導体多層コンプトンカメラの実現には、半導体検出器だけでなく、その読み出しシステムも不可欠である。そのため、検出器開発と同時に読み出しシステムの研究、開発も進めて行く必要がある。しかしながら、DSSD のような非常に多チャンネルの信号出力を持った検出器読み出しシステムの実現は容易ではなく、検出器そのものの原理、基礎特性や、読み出しのためのアナログ集積回路、そして読み出されたデータを処理するためのディジタル回路など、広範囲に渡る物理学的、技術的な知識が必要となってくる。これらを全て習得するには長い時間が必要であるため、本研究では将来、DSSD の読み出しシステムの開発を進める際に必要な、半導体検出器の多チャンネル読み出しの基礎知識、基礎技術の習得を目指す。そのために、DSSD の検出原理やコンプトンカメラの原理、現状の多チャンネル読み出しシステムシステムについて理解し、新しく導入された 4 cm DSSD の立ち上げを行った。そして、間もなく導入される予定の新しい読み出しシステムの概要について理解し、新しい読み出しシステムの立ち上げに備えた。さらに、将来読み出しシステムが衛星に搭載される場合に対応しなければならないため、衛星上での装置間統一通信規格である SpaceWire について作業を行った。具体的には、SpaceWire に対応した読み出しシステムの開発を進める上で必要となるデバイスである FPGA と、ハードウェア記述言語である VHDL の基礎を習得するため、SpaceWire を搭載した ADC ボード上の FPGA に独自に設計した回路を組込み、マルチチャンネルアナライザを開発した。

第2章　DSSD

2.1　半導体検出器

X 線・軟ガンマ線を検出するには、主にシンチレーション検出器と半導体検出器が用いられる。シンチレーション検出器とは、CsI や BGO などといった物質の結晶中に放射線が入射した際、放射線と物質との相互作用により結晶が蛍光を発することを利用した検出器である。蛍光の強さは放射線の失ったエネルギーに比例し、光電子増倍管 (PMT) やフォトダイオード (PD) などを用いて電気信号に変換される。シンチレーション検出器は阻止能が大きいといった利点を持つが、エネルギー分解能はあまり良くない。

半導体検出器は、シリコンやゲルマニウムなどを用いた検出器で、シンチレーション検出器に比べて良いエネルギー分解能を得ることができる。半導体検出器は p 型半導体と n 型半導体が接合したダイオードの構造を持っており、接合付近にはキャリアがほとんど存在しない空乏層と呼ばれる領域がある。空乏層に放射線が入射すると、X 線やガンマ線の場合には検出器の物質との間に光電吸収、コンプトン散乱、電子対生成の 3 種類の相互作用を起こす。これらの相互作用により、物質中の束縛電子がエネルギーを受けとるなどして高エネルギーの電子が作られ、この高エネルギー電子が空乏層内に電子・正孔対を生成する。生成された電子・正孔対を電荷信号として取り出すことにより、X 線やガンマ線を検出することができる。このように、半導体検出器では空乏層が放射線の検出領域となる。一対の電子・正孔対を生成するのに必要なエネルギーは、シリコンではおよそ 3.65 eV である。そのため、検出器から出力される電荷量は検出器内で失われた放射線のエネルギーに比例する。ただし、このままの状態では生じた電子・正孔対が捕獲や再結合により消滅しやすく、電荷収集が不完全になりやすい。また、空乏層の厚さが極めて小さく、検出器の持つ静電容量が大きくなるため、前置増幅器に接続したときの雑音特性が非常に悪くなる。そこで、通常半導体検出器にはダイオードを逆バイアスする方向の電圧をかけて使用する。逆バイアスをかけることにより検出器内には新たな電場が生じ、キャリアが電極側に移動して空乏層が厚くなり静電容量が減少し、放射線により生じた電子・正孔対を速やかに分離、収集することができるようになる。図 2.1 に半導体検出器の原理を示す。

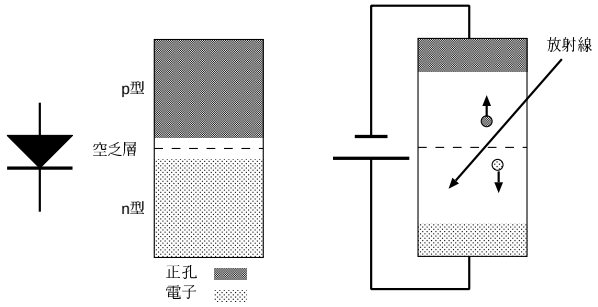


図 2.1: 半導体検出器の原理

2.2 シリコンストリップ検出器

2.2.1 シリコンストリップ検出器の構造

シリコンストリップ検出器 (Silicon Strip Detector: SSD) は、もともと加速器実験の分野で短命粒子の検出のために開発されたものである。SSD は、n 型のシリコンウェハー上に短冊状の p+領域を並べたもので、多数の細長いダイオードを形成し、それぞれのダイオードが独立した検出器として働く。放射線によって生成された電子・正孔対のうち、正孔がもっとも近い p+領域に収集される。ストリップの間隔は数 10〜数 100 μm であるため、放射線の 1 次元での反応位置を高い精度で測定できる。また、SSD のもう一つの特徴として、高速応答性があげられる。典型的な 300 μm 厚の SSD の場合、その deadtime は 30 ns 程度で、プラスチックシンチレータに匹敵する速さである。

p+のストリップだけでなく、その裏側に n+のストリップを形成したものを、両面シリコンストリップ検出器 (Double-sided Silicon Strip Detector: DSSD) と呼んでいる。図 2.2 に DSSD の基本的な構造を示す。

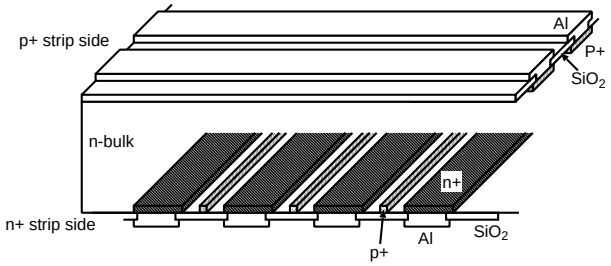


図 2.2: 基本的な両面シリコンストリップ検出器の構造 [11]

図 2.2 から分かるように、n 側ストリップは p 側ストリップと直交するように配置されている。DSSD では正孔に加え、n+ストリップで電子を収集し、放射線の反応位置を 2 次元で知ることができる。

2.2.2 リーク電流

半導体検出器に逆バイアスをかけると、数 nA 程度のわずかな電流が流れる。この電流をリーク電流と呼ぶ。逆バイアスは p 型領域と n 型領域の接合面から多数キャリアを引き離す方向にかかっているが、少数のキャリアは引き合うため、接合を越えて電流が流れる。ただし、通常この原因による電流は小さく、重要なリーク電流源になることはまれである。

リーク電流の主要な発生源は、空乏層内の電子・正孔対の熱励起によるものである。熱励起された電子や正孔がエネルギーギャップを越えて伝導帯に遷移し、電流が流れる。また、検出器に用いられる半導体結晶は非常に高純度なものであるが、極わずかに存在する不純物により、エネルギーギャップの中に新たなエネルギー順位がつくられ、電子・正孔を伝導帯に遷移しやすくしていることもリーク電流を増やす原因となる。熱励起によるリーク電流は温度に依存し、一般に

$$I(T) \propto \exp\left(\frac{-E}{2k_B T}\right) \tag{2.1}$$

T : 温度

B : エネルギーギャップ

k_B : ボルツマン定数

で表される。熱励起によるリーク電流は温度を下げれば大きく減少するため、検出器を低温下で動作させることで十分に減らすことができる。リーク電流はこの他にも、放射線による検出器の損傷によって増えることが知られており、検出器の損傷の度合を知るのに役立つこともある。

半導体検出器は非常に小さな信号を扱うため、リーク電流による回路系雑音が問題になりやすい。また、検出器に接続される前置増幅器の動作にも悪影響を及ぼすこともある。そのため、リーク電流はできる限り小さいことが望ましい。

2.2.3 エネルギー分解能

半導体検出器のエネルギー分解能は、放射線により生じる電子・正孔対の数の統計揺らぎと電子回路系のノイズに依存する。一対の電子・正孔対を作るのに消費される放射線のエネルギーは電離エネルギー ϵ と呼ばれ、前述したようにシリコンの場合は約 3.65 eV である。放射線が失ったエネルギーを E とすれば、生成された電子・正孔対の全数は $\frac{E}{\epsilon}$ である。電子・正孔対の生成がポアソン過程であると仮定した場合、電子・正孔対の全数の分散は全数 $\frac{E}{\epsilon}$ に等しくなる。しかし、実際に測定される分散の値はこれよりも小さい。このずれを調整するためにファノ因子 F が導入されており、

$$F \equiv \frac{\text{統計分散の観測値}}{E/\epsilon} \tag{2.2}$$

で定義される。ファノ因子はシリコン検出器の場合約 0.1 である。よって、統計揺らぎに依存するエネルギー分解能を半値幅 (FWHM) で表すと、

$$\Delta E_{stat} = 2.355\epsilon\sqrt{F\frac{E}{\epsilon}} \tag{2.3}$$

となる。

次に回路系のノイズであるが、まず DSSD などの半導体検出器から前置増幅器 (CSA) までの等価回路を図 2.3 と図 2.4 に示す。ここで、C_d は検出器の持つ容量である。DC 結合とは検出器と CSA を直接接続する方式で、AC 結合はコンデンサを通して検出器と CSA を接続する方式である。バイアス電圧が低い場合などは DC 結合が用いられるが、高いバイアス電圧をかける場合は AC 結合が用いられる。また、AC 結合ではリーク電流が CSA に流れ込むのを防ぐこともできる。DSSD では p+側が DC 結合で、n+側はバイアスをかけるために AC 結合となっている。DSSD で AC 結合をするには、RC-chip が用いられる。これは、カップリングコンデンサとバイアス抵抗をシリコン基板上に作成したものである。RC-chip はその構造上、カップリングコンデンサ以外にも容量 C_{rc} を持っている。

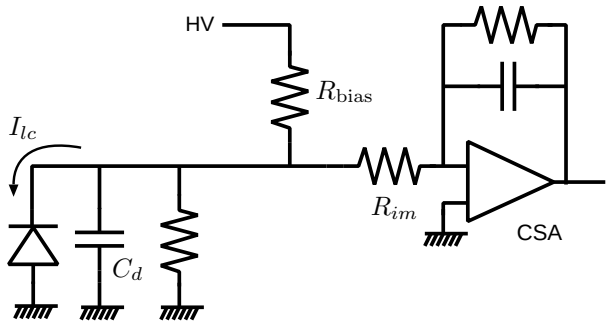


図 2.3: DC 結合の等価回路

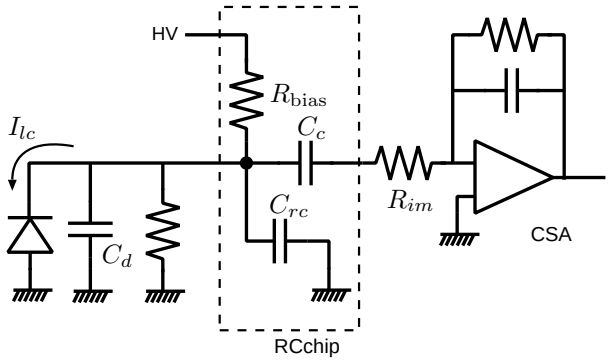


図 2.4: AC 結合の等価回路

回路系ノイズには、ジョンソンノイズ、ショットノイズ、そして CSA で生じるノイズがある。ジョンソンノイズとは、検出器を含めた電子回路内に存在する直列の抵抗成分において、その中でキャリアが不規則に熱運動することにより生じる雑音である。そのため、ジョンソンノイズは熱雑音とも呼ばれている。半導体検出器の信号は電荷として出力されるため、ノイズの大きさを電荷で表した等価雑音電荷 (Equivalent Noise Charge: ENC) が用いられる。DSSD の読み出しに VA32TA と呼ばれるチップを用いた場合、バイアス抵抗 R_{bias} で生じるノイズ ENC_{bias} は、経験

的に

$$ENC_{\mathrm{bias}}(\mathrm{e}^-) = 770\sqrt{\frac{\tau(\mu\mathrm{s})}{R_{\mathrm{bias}}(\mathrm{M}\Omega)}} \tag{2.4}$$

で計算される。ここで τ は波形整形の時定数である。また、回路内に存在する他の抵抗成分 R_{im} によってもジョンソンノイズが発生しており、

$$ENC_{im}(\mathrm{e}^-) = 450\frac{C_d\sqrt{R_{im}(\mathrm{M}\Omega)}}{\rho\sqrt{\tau(\mu\mathrm{s}}} \tag{2.5}$$

と計算される。ここで、ρ は電荷収集効率である。

次にショットノイズとは、検出器のリーク電流 I_{lc} によるキャリアの流れが統計的に揺らぐことによって生じるノイズで、ジョンソンノイズと同様

$$ENC_{lc}(\mathrm{e}^-) = 107\sqrt{I_{lc}\tau(\mathrm{nA}\cdot\mu\mathrm{s})} \tag{2.6}$$

で計算される。

最後の CSA で生じるノイズであるが、これは CSA にロードされる容量に依存し、DSSD と VA32TA の組合せでは、

$$ENC_{\mathrm{amp}}(\mathrm{e}^-) = \frac{45 + 19\times C_{in}(\mathrm{pF})}{\rho\sqrt{\tau(\mu\mathrm{s})}} \tag{2.7}$$

となる。ここで、C_{in} は CSA にロードされる容量である。DSSD は図 2.5 に示すように、p 側ストリップと n 側ストリップの間に存在する Body Capacitance と、隣接するストリップの間に存在する Inter-strip Capacitance と呼ばれる容量を持っている。よって、C_d は Body Capacitance と

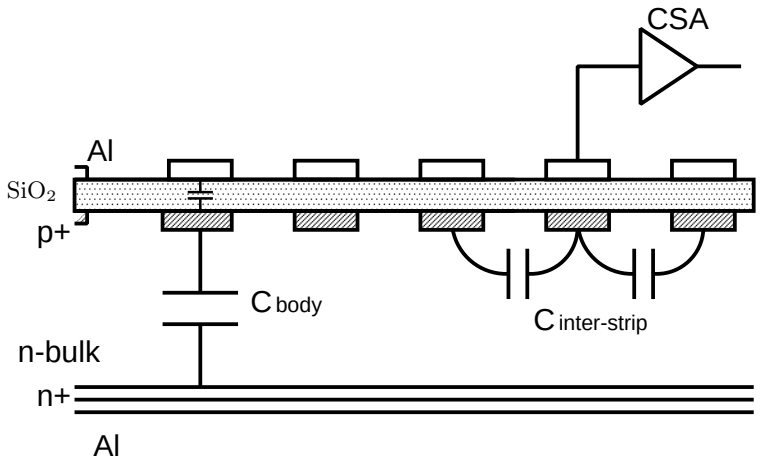


図 2.5: DSSD が持つ容量 [11]

Inter-strip Capacitance を合成した容量となる。また、AC 結合の場合、RC-chip の容量 C_{rc} が加わるため、$C_{in} = C_d + C_{rc}$ となり、DC 結合よりもノイズが増加する。電荷収集効率 ρ は、DC 結合の場合は 1 であるが、AC 結合の場合は

$$\rho = \frac{C_c}{C_c + C_d + C_{rc}} \tag{2.8}$$

となる。通常、AC 結合する際はカップリングコンデンサの容量を数 1000pF 程度と十分大きくし、電荷収集効率を 1 に近くできるが、RC-chip に作ることのできるコンデンサの容量が数 10pF 程度しかなく、他の容量が無視できなくなり、DSSD の n 側での電荷収集効率が低くなっている。そのため、n 側でのノイズをさらに増加させる要因となっている。

ここまでの全てのノイズを考慮した E_{total} は各ノイズ成分の二乗平均となり、

$$ENC^2_{\mathrm{total}} = ENC^2_{\mathrm{bias}} + ENC^2_{im} + ENC^2_{lc} + ENC^2_{\mathrm{amp}} \tag{2.9}$$

となる。よって、統計揺らぎとノイズを考慮したエネルギー分解能を半値幅で表すと

$$\Delta E(FWHM) = 2.355\epsilon\sqrt{\left(F \cdot \frac{E}{\epsilon}\right) + ENC^2_{\mathrm{total}}} \tag{2.10}$$

となる。

2.3 半導体多層コンプトンカメラと NeXT 衛星

DSSD の高い位置分解能と、エネルギー分解能を応用して、硬 X 線から軟ガンマ線領域での高感度な観測を行うコンプトンカメラと呼ばれる観測装置が開発されている。

2.3.1 コンプトンカメラ

硬 X 線から軟ガンマ線の領域では、図 2.6 に示すように、光子と物質との相互作用はコンプトン散乱が支配的である。そのため、従来の検出器ではこのエネルギー領域での感度の高い観測が難しかった。そこで、コンプトン散乱を積極的に利用するコンプトンカメラが考案された。

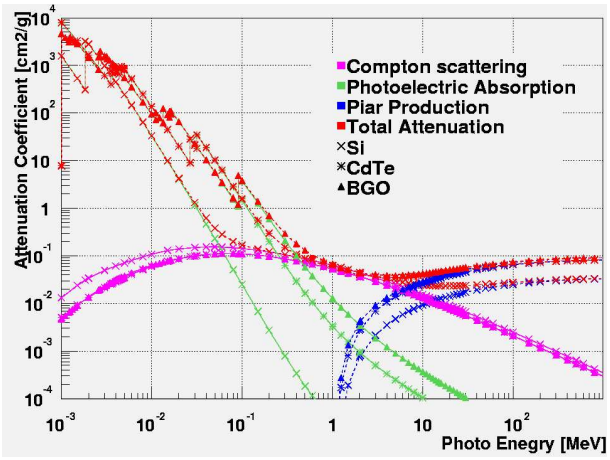
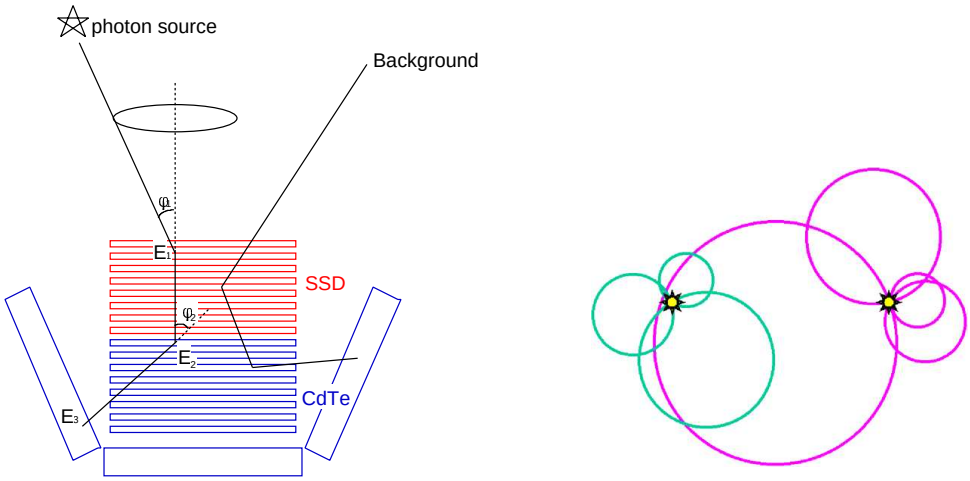


図 2.6: 吸収係数 [14]

ガンマ線が検出器に入ると、コンプトン散乱により持っていたエネルギーの一部を電子に与え、検出器から出ていく。コンプトンカメラでは、検出器でガンマ線光子が落としたエネルギーと、そ

の反応位置を測定することで、コンプトン散乱を再構成し、入射したガンマ線光子が持っていたエネルギーと到来方向を知る。コンプトンカメラは図 2.7(a) のようにシリコンやテルル化カドミウム (CdTe) などの半導体検出器を数十層にわたって積み重ねた構造をしており、内部でガンマ線光子を複数回コンプトン散乱させた後、光電吸収させる。



(a) コンプトンカメラの概念図

(b) 複数イベントの重ね合わせ

図 2.7: コンプトンカメラ [11]

入射したガンマ線がコンプトン散乱により検出器に落としたエネルギーを E_1, E_2、散乱された角度を ϕ_1, ϕ_2、光電吸収されたエネルギーを E_3 とすると、エネルギー保存則と運動量保存則から、

$$\cos\phi_1 = 1 + \frac{m_e c^2}{E_1 + E_2 + E_3} - \frac{m_e c^2}{E_1 + E_2} \tag{2.11}$$

$$\cos\phi_2 = 1 + \frac{m_e c^2}{E_2 + E_3} - \frac{m_e c^2}{E_3} \tag{2.12}$$

となる。ここで、m_e は電子質量、c は光速である。これにより、ガンマ線の到来方向を頂角 ϕ_1 の円錐面に制限できる。さらに複数のイベントを重ね合わせることにより、ガンマ線の到来方向を一点に絞ることができ、バックグラウンドの除去も可能となる。

コンプトンカメラの角分解能は検出器の位置分解能の他に、エネルギー分解能に大きく依存する。また、検出器内の電子は実際には静止しておらず、持っている運動量が不定であるために、散乱角が広がりを持ってくる。この効果は Doppler Broadening と呼ばれ、検出器の原理上避けることはできない。半導体物質では比較的 Doppler Broadening が小さく、その中でも Si が最も有利である。コンプトンカメラでは、エネルギー分解能による角分解能の劣化を Doppler Broadening による影響に対して十分に下げておくことが求められる。

2.3.2 NeXT 衛星

現在、JAXA/ISAS を中心に、すざく衛星に続く次期 X 線天文衛星 NeXT(New X-ray Telescope) 計画が進められている。NeXT 計画には国内外の多数の大学、研究機関が参加しており、2011 年の打ち上げを目指している。

NeXT 衛星には、コンプトンカメラを用いた軟ガンマ線検出器 (Soft Gamma-ray Detector: SGD) の搭載が検討されている。SGD はコンプトンカメラにすざく衛星の HXD で実現された BGO による井戸型シールドとファインコリメータを加えた、「狭い視野を持つ半導体多層コンプトン望遠鏡」という新しい概念の検出器である。シールドによるバックグラウンドの除去にコンプトンカメラによるバックグラウンドの除去が加わり、10-300 keV で過去最高の感度を得る。これにより、銀河団などからの非熱的な放射を、初めて正確に測定できるものと期待されている。また、SGD ではコンプトン散乱の原理を用いて、100 keV という高いエネルギー領域で初めての偏光観測が可能となる。偏光は粒子が加速される環境での磁場やジオメトリの情報を与えるが、今までは良い観測手段が無かった。

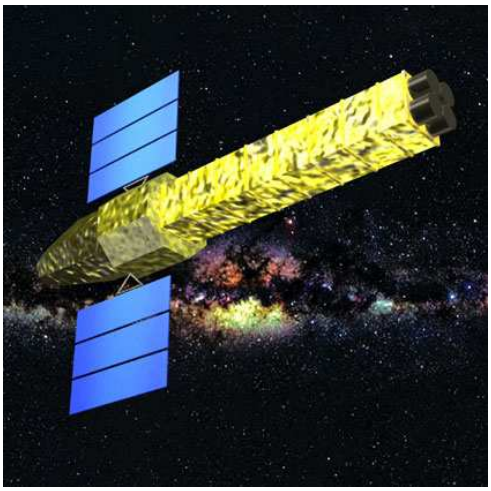


図 2.8: NeXT 衛星の予想図 [14]

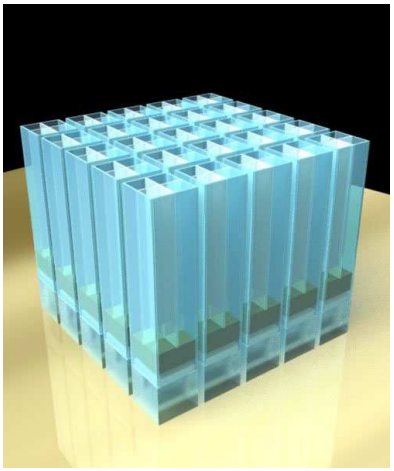


図 2.9: SGD の予想図 [14]

第3章　VA32TAとVA-DAQを用いたDSSD多チャンネル読み出しシステム

3.1　VA32TA

半導体検出器からの信号を読み出す場合、CSA、Shaping Amplifier、Discriminator などが必要である。DSSD ではチャンネル数が多くなるため、これらの単体装置を複数用意することは、スペース、消費電力などの点で現実的ではない。そこで従来より、読み出しに必要な機能を一つのチップにまとめた LSI が開発されてきた。VA32TA はノルウェーの IDE 社、スタンフォード線形加速器センター (SLAC)、JAXA/ISAS により、従来加速器実験に使われていたものを DSSD の読み出しのために最適化して開発されたもので、32ch 分の CSA、Shaping Amplifier、Discriminator、Sample Hold 回路、マルチプレクサ、トリガ出力を持ったアナログ LSI である。VA32TA は 0.35 μm の CMOS プロセスで製造され、大きさは $7.035 \times 3.385 \times 0.725$ mm と小型である。また、1 チャンネルあたりの消費電力は 6.3 mW と低消費電力である。

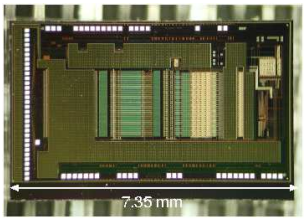


図 3.1: VA32TA チップ [11]

3.1.1　VA32TA の特徴

VA32TA は VA part と TA part の 2 つの部分に分けることができる。図 3.2 に VA32TA のブロック図を示す。VA part では、CSA、slow shaper、Sample Hold 回路により波形整形が行われ、TA part では、fast shaper、Discriminator によりトリガ信号が生成される。

検出器と DC 結合した際は、検出器のリーク電流が CSA に流れ込んで問題となることがある。VA32TA ではリーク電流と同じ大きさで逆向きの電流を CSA のフィードバック抵抗に流すことで、リーク電流を打ち消す電流補償機能が搭載されている。また、VA32TA 内部の各回路が動作するのに必要なバイアスは、外部から直接 VA32TA のパッドを通して供給することもできるが、チップ内部でつくり出すこともできる。各バイアスの設定および、電流補償機能の設定や、トリガをマスクするチャンネルの設定は、VA32TA 内部の 199 bit のシフトレジスタに書き込むことで

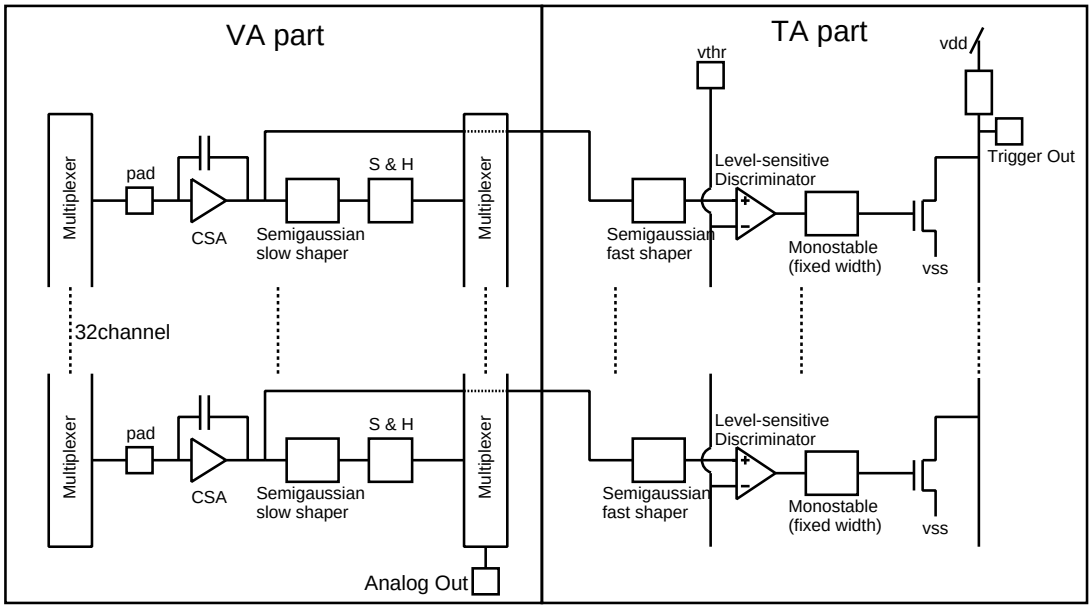


図 3.2: VA32TA のブロック図 [11]

設定している。宇宙空間でチップを使用する場合、放射線によってレジスタが保持している値が反転してしまうことがあるが、VA32TA では 3 つのレジスタに同じ値を保持しておき、多数決をとることによって信頼性を確保している。199 bit のうち数 bit づつが各バイアスの設定に割り当てられており、この設定値をもとに内部の DAC がバイアスを生成している。

3.1.2 VA32TA の読み出しシーケンス

図 3.3 に VA32TA の読み出しタイミングチャートを示す。DSSD から出力された信号は、ワイヤーボンディングにより VA32TA のパッドに入力される。入力された信号は CSA に入り、そこで電荷から電圧の信号に変換される。CSA の出力は VA part と TA part へと分岐される。TA part に入った信号は fast shaper によって波形整形された後、Discriminator により vthr 電圧と比較され、スレッショルドを越えたチャンネルからトリガ信号が出される。トリガ信号は全チャンネルで OR され、チップからのトリガ信号 (ta,tb) として外部に出力される。また、VA part に入った信号は slow shaper によって波形整形される。slow shaper の時定数は fast shaper よりも遅いため、トリガ信号が出力されてから波形のピークがくることになる。VA32TA 外部の制御回路により、チップのトリガが立ってから一定時間 (peaking time) 後に holdb 信号をチップに入力することで、その時点での波高が Peak Hold される。次に shift_in_b 信号をチップに入力することで読み出しが開始される。Clkb 信号に合わせて内部のマルチプレクサが順次切り替わり、analog out に Peak Hold されたアナログ値が出力される。32ch 分の出力が終わると shift_out_b 信号が出力される。

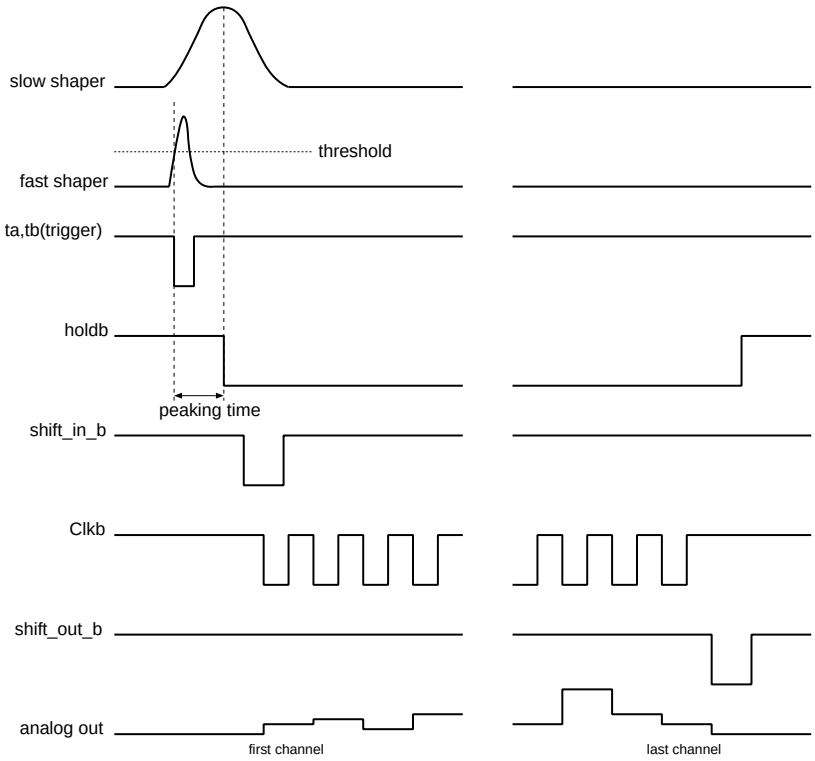


図 3.3: VA32TA の読み出しタイミングチャート

3.2 VA-DAQ

VA-DAQ とは、VA32TA の制御と読み出しを行うためのシステムで、IDE 社により提供されていた。VA-DAQ はパラレルポートを介して PC と接続し、LabView による制御ソフトウェアが付属している。このソフトウェアにより、グラフィカルな画面を見ながら VA32TA のパラメータを設定したり、データの収集を行なうことができる。

図 3.4: VA-DAQ [11]

3.2.1 VA-DAQ の機能

VA-DAQ 内部は、PARIO、SUPPLY、BIAS、ADC、PULSE と呼ばれる 5 つの部分に分けることができる。PARIO は PC とパラレルポートで接続するためのインターフェイスである。SUPPY は VA32TA の電源である+1.5 V と-2.0 V を生成する。BIAS は VA32TA に必要なバイアスを生成する。ADC は分解能 14 bit の ADC で、VA32TA のアナログ出力や、各種バイアスのモニタ値を AD 変換する。また、PULSE は 199 bit のシフトレジスタに値を書き込む際のディジタル信号や、チップの調整に使うテストパルスを生成する。

3.2.2 VA-DAQ を用いた多チャンネル読み出し

DSSD では片面あたり数十チャンネルの信号を読み出す必要がある。32ch 以上読み出す場合は、VA32TA を複数使わなければならない。VA32TA を複数使った場合の VA-DAQ との接続を図 3.5 に示す。VA32TA は FEC(Front End Card) と呼ばれる基板に 1 個または 2 個ずつ搭載されている。そして、FEC と VA-DAQ は Adapter Card と呼ばれる基板を通してフラットケーブルで接続されている。また、shift_in_b および shift_out_b 信号以外の信号や電源は全てのチップで共有されている。読み出し開始を伝える shift_in_b 信号は、まず VA-DAQ から出力され、最初の VA32TA に入る。そして、最初のチップの読み出しが終了したときに出される shift_out_b 信号が次のチップの shift_in_b 信号となる。shift_out_b が次々と次のチップの shift_in_b に入力されるように Adapter Card 上のジャンパーを設定することにより、全チップの読み出しが行われる。この間 VA-DAQ は全チャンネル数分の Clkb 信号を出していれば次々と波高のアナログ値が送られてくることになる。

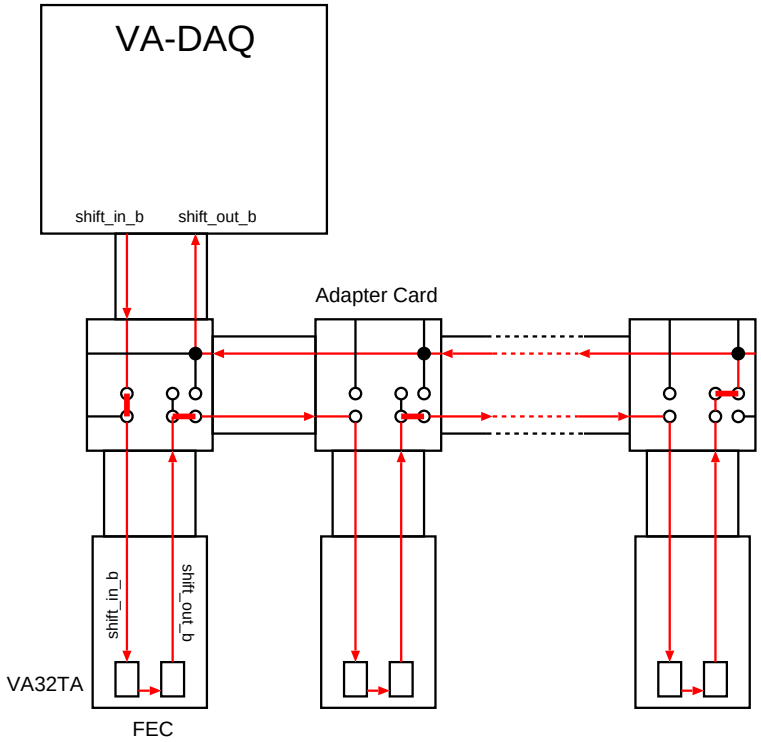


図 3.5: VA32TA と VA-DAQ の接続

3.3 読み出しの実際

3.3.1 4 cm DSSD でのセットアップ

現在、これまでで最も面積の大きい 4 cm の DSSD が製作され、M1 の安田さんにより評価が行なわれている。この 4 cm DSSD の立ち上げを、安田さんと共に行なった。

図 4.2 に 4 cm DSSD の写真を示す。4 cm DSSD は片面あたり 96 チャンネルのストリップがあるため、VA32TA を 3 個使って読み出している。写真では n 側が上になっているため、DSSD と FEC の間に RCchip が見える。また、図 4.3 にセットアップ全体の写真を示す。DSSD には 100 V のバイアス電圧をかけるが、RCchip の耐圧が 100 V 無いため、100 V と 120 V の 2 つのバイアス用いることで、RCchip にかかる電圧を 20 V におさえている。また、RCchip 上のバイアス抵抗とは別に、10 MΩ のバイアス抵抗を外部に設けてある

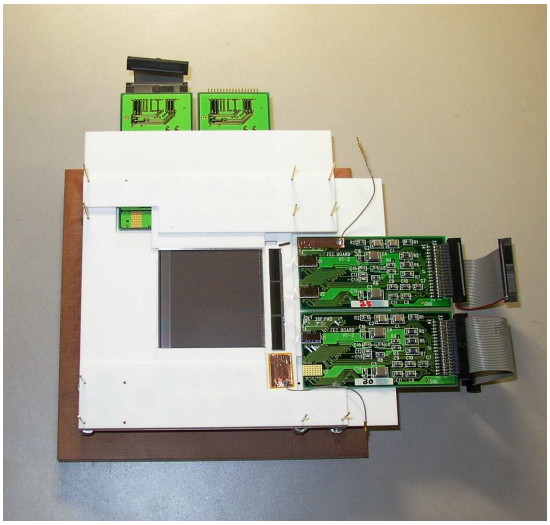


図 3.6: 4 cm DSSD

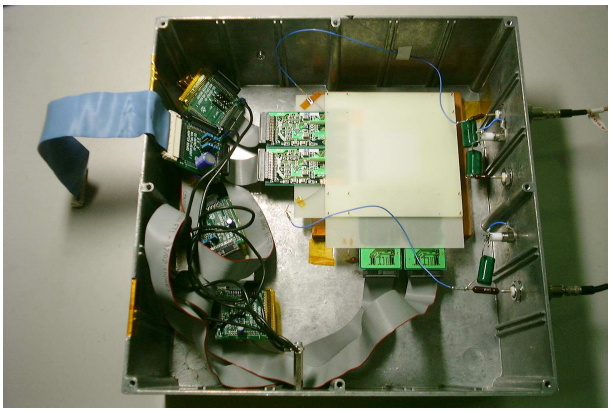


図 3.7: セットアップ

3.3.2 解析方法

解析には、CERN で開発されている ROOT と呼ばれるソフトウェアを使用した。ROOT はオブジェクト思考を採り入れた高エネルギー物理学用の解析ソフトウェアであり、C++言語のインタプリタを内蔵している。そのため、C++言語で作成したプログラムをマクロとして実行できる。また、C++プログラムをマクロとして実行するだけでなく、C++のライブラリとして ROOT を用いることもでき、ROOT は一種のアプリケーションフレームワークでもある。DSSD の解析に用いたプログラムは、始めはマクロとして実行していたが解析に非常に時間がかかるため、コンパイルして独立のアプリケーションとしたところ、およそ 4 倍の速さで実行できるようになった。

VA-DAQ で得られたデータは、AD 変換された波高値を最初のイベントの 1 番目のチャンネルから順番に並べていったバイナリ形式のデータである。そのままでは扱いにくく、p 側と n 側では符号が逆である。そのため、まずは扱いやすいデータ形式に変換し、p 側と n 側の符号をそろえる操作を行なう。この操作をすることで、データは ROOT の Tree と呼ばれるクラスを用いて格納され、以降の解析で扱いやすくなる。

ペデスタル補正

DSSD から得られるデータでは、一つ一つのチャンネルでゼロ点からのずれ (ペデスタル) が存在する。これは、VA32TA 内部のトランジスタの特性の違いや、DSSD のそれぞれのチャンネルのリーク電流の違いなどによって VA32TA の出力にオフセットが乗ることによって生じる。そのため、まずペデスタルを補正する必要がある。これは、各チャンネルについて、10000 イベントの平均を求め、それをペデスタルとして各チャンネルの全てのデータから引くことで補正する。

コモンモードノイズの除去

あるチャンネルでイベントが起きると、その影響で全チャンネルに揺らぎが生じる。この揺らぎは全チャンネルで共通だと考えられるため、コモンモードノイズと呼ぶ。まず、最も波高の高いチャンネル 2 つと、最も波高の低いチャンネル 2 つを除いたチャンネルでデータの平均値をとり、これをコモンモードノイズとする。そして、イベントごとに全チャンネルから引くことでコモンモードノイズを除去する。

以上のような操作で、一応は各チャンネルのスペクトルを得ることができる。ただし、このとき得られるスペクトルは波高値 (ADC channel) のヒストグラムであるため、X 線やガンマ線のエネルギーのスペクトルを得るためには線源のピークのエネルギーに相当する ADC channel を求め、ADC channel とエネルギーを変換する必要がある。また、それぞれのチャンネルではエネルギーと ADC channel を変換するときのゲインが異なるため、同じエネルギーでもピークの ADC channel は異なる。そのため、加算スペクトルを作ったりする場合は、ピークの値を用いて各チャンネルのゲインを補正する必要がある。

3.3.3 測定結果の例

セットアップした 4 cm DSSD で実際に放射線源の測定を行なった。測定に使用した線源は ^{241}Am で、59.5 keV の X 線を出す。

測定は DSSD に 100 V のバイアス電圧をかけ、恒温槽で-10 ℃に冷却して行なった。また、DSSD の場合放射線が入ると、 p 側と n 側両方から同時に信号が出力されるため、どちらか一方のストリップでトリガをかければ良いが、この場合はノイズ特性の良い p 側だけでトリガをかけた。

図 3.8 と図 3.9 に、ペデスタル・コモンモードノイズの補正をする前と補正した後の 1 ストリップ分 (p 側) のスペクトルを示す。スペクトルの 0ch 付近のピークは、このストリップ以外のストリップでトリガがかかったときに得られたデータで、設定した threshold に関係なく出現する。図 3.10 は n 側ストリップで得られたスペクトルで、p 側と比べると明らかにエネルギー分解能が悪いことが分かる。そして、図 3.11 は p 側ストリップ全てについて、59.5 keV のピークをガウシアン関数でフィッティングし、エネルギー分解能 (FWHM) を求めたものである。この測定は立ち上げ直後に行なったもので、まだあまりエネルギー分解能は良くないが、その後安田さんにより VA32TA のパラメータの最適化が行なわれ、平均しておよそ 1.9 keV(FWHM) のエネルギー分解能を達成している。

DSSD は 2 次元の反応位置の情報を取得することができるため、X 線やガンマ線でのイメージを得ることができる。イメージを作成するには、p 側ストリップと n 側ストリップ両方で同時に反応しているイベントで、なおかつイメージを作りたいエネルギー範囲に対応する波高値の範囲に含まれているイベントを選び出し、そのイベントで反応している p 側ストリップと n 側ストリップの位置を x 座標、y 座標として 2 次元ヒストグラムを作成する。このようにして得られたイメージを図 3.12 に示す。線源を DSSD のカバー上に直接置いて測定したため、線源は DSSD から数 mm しか離れておらず、線源を置いた位置を中心に等方的に X 線が出ている様子を見ることができる。

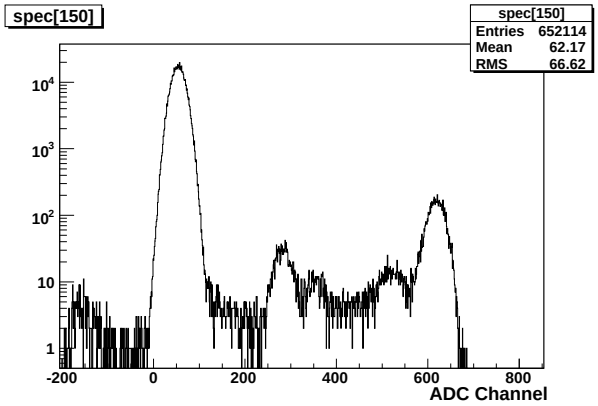


図 3.8: p 側ストリップのペデスタル・コモンモードノイズ補正前のスペクトル

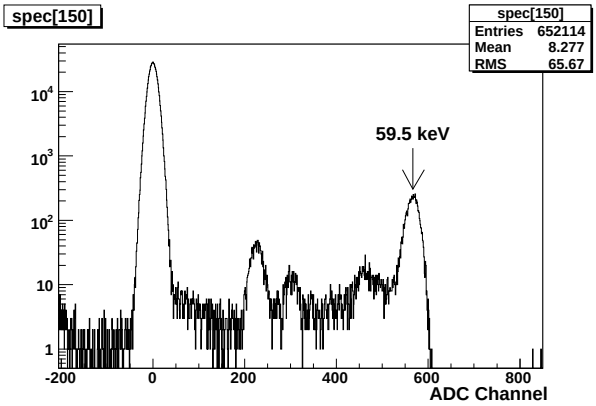


図 3.9: p 側ストリップのペデスタル・コモンモードノイズ補正後のスペクトル

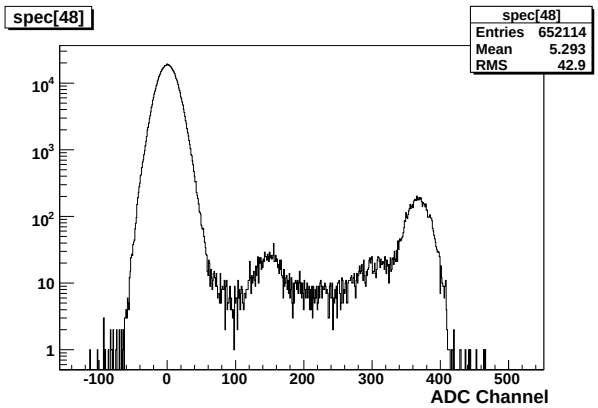


図 3.10: n 側ストリップのスペクトル (補正後)

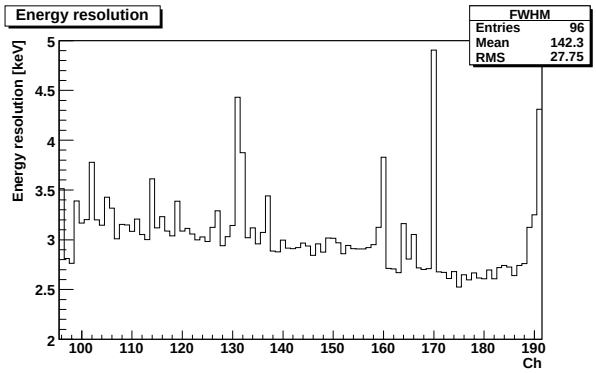


図 3.11: p 側ストリップのエネルギー分解能

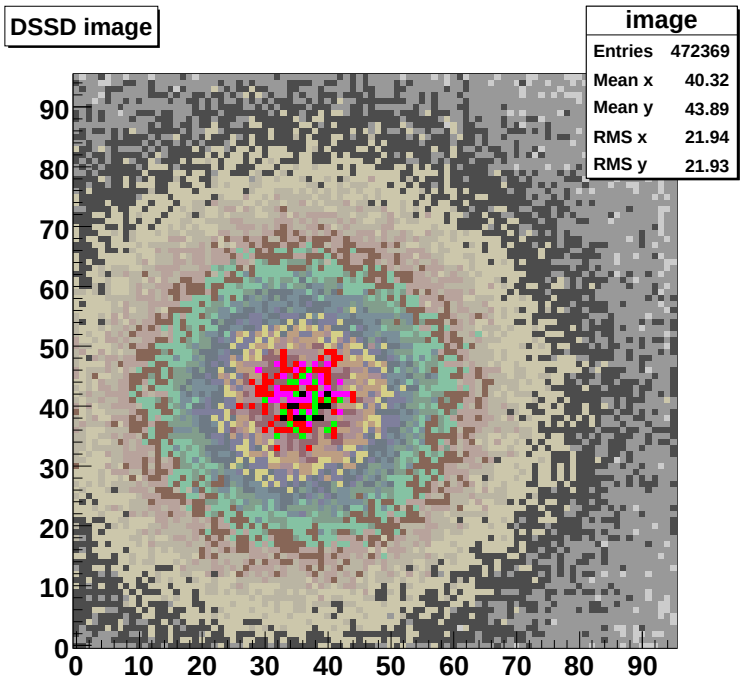


図 3.12: 線源のイメージ

3.4 VA-DAQ の問題点

VA-DAQ は LabView を用いた制御ソフトウェアにより、操作が容易であるという利点はあるが、ハードウェア面、ソフトウェア面共に以下のような問題点がある。

まずハードウェア面では、将来衛星に搭載することを考えた場合、VA-DAQ の消費電力、大きさ、外部とのインターフェイスが問題となり、そのまま搭載することはできない。パラレルポートのデータ転送速度も 500 kB/s 程度と遅く、DSSD を多層化してコンプトンカメラを構成した場合、その読み出しチャンネルの多さに対応できなくなる。また、後述のフローティングをすればRC-chip を省くことができるのだが、VA-DAQ はフローティングに対応していない。さらに、4 cm DSSD の立ち上げ中に問題となったものに、制御 PC からノイズが入りやすいといったこともある。4 cm DSSD では当初ノイズが下がらず、DSSD の評価を行なうことができないでいたのだが、制御用 PC の筐体を VA-DAQ 等他の機器と共通の GND に落としたところ、ノイズが劇的に低下した。PC と VA-DAQ はパラレルポートを用いて接続されるが、PC と VA-DAQ が電気的に絶縁されていないため、PC のノイズの影響を大きく受けてしまう。通常、微弱な信号を扱う計測機器では PC とのインターフェイス部分は電気的に絶縁されていることが望ましい。

次にソフトウェア面では、制御ソフトが OS に依存しているといった問題がある。LabView 自体は Windows のバージョンに依存していないのだが、LabView に埋め込まれているコードに Windows 9x でしか実行できないものが含まれており、制御 PC を更新しようとした場合に問題となる。

そして、ハードウェア、ソフトウェア両面に言えることとして、VA-DAQ は企業から提供されているものであるため全ての情報が公開されておらず、ブラックボックスとなっている部分がある。そのため、何か問題が発生したり、改良を加えようとした場合に情報が無くて困る場合がある。また、VA-DAQ の生産は既に終了しており、VA-DAQ を用いた読み出しシステムのこれ以上の発展は難しい。

3.5 新型 DSSD 読み出しシステム

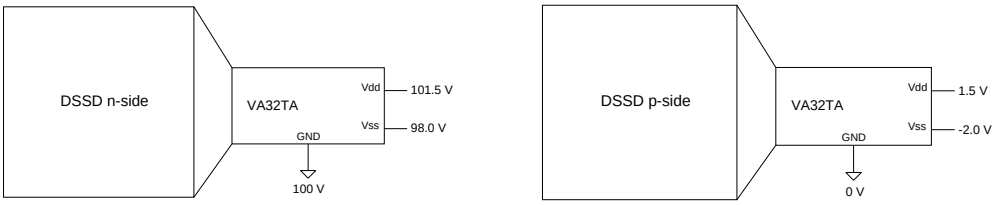
現在、VA-DAQ に代わる新しい DSSD 読み出しシステムが、JAXA/ISAS およびクリアパルスによって開発されている。新しいシステムはフローティングに対応しており、RC-chip を用いない DSSD の読み出しが可能である。また、衛星搭載を考慮し、将来的には SpaceWire で読み出しを行なうことも可能である。

3.5.1 フローティング

DSSD にバイアス電圧をかける場合、VA32TA に高いバイアス電圧がかからないように、RC-chip を用いて AC 結合にしなければならない。しかし、前述のように RC-chip を使うと電荷収集効率が悪くなり、n 側ストリップのエネルギー分解能が悪化する。また、RC-chip の耐圧の関係上、やや複雑なバイアスのかけ方をしなければならず、セットアップが繁雑になっている。

そこで、RC-chip を用いずにバイアスをかけることを可能にする、フローティングと呼ばれる手法が用いられる。VA-DAQ を用いた現在のシステムでは、VA32TA やその他の回路、バイアスの GND は全て共通でなければならないが、VA32TA 自身は必ずしも後段のディジタル回路と GND を共通にする必要は無く、VA32TA の GND を他の回路から浮いた状態にすること (フローティング) も可能である。VA32TA は GND の電位が他の回路と比べてどうであろうと、電源電圧 (Vdd = 1.5 V, Vss = -2.0 V) や入力信号のレベルさえ正しければ正常に動作する。そこで、n 側ストリップを読み出す VA32TA の GND を、バイアスのかかった n 側ストリップと同じ電位にすれば、RC-chip を省き、DC 結合によって n 側ストリップを読み出すことが可能になる。

図 3.13 にフローティングをしたときの DSSD と VA32TA の模式図を示す。仮に DSSD に 100 V のバイアス電圧をかけるとすると、n 側ストリップに接続される VA32TA の GND の電位を、バイアスの GND に対して 100 V にすれば良い。この場合、VA32TA の電源としては、Vdd に 101.5 V を、Vss に 98.0 V をかければ良い。また、p 側ストリップに接続される VA32TA は現在の読み出しシステムと同じく、GND に 0 V を、Vdd と Vss にそれぞれ 1.5 V と-2.0 V をかければ良い。



(a)n 側ストリップ　　(b)p 側ストリップ

図 3.13: フローティング

3.5.2 新型読み出しシステムの特徴

図3.14に新しい読み出しシステムのブロック図を示す。新しい読み出しシステムでは、FEC(Front End Card) に VA32TA が 2 枚または 3 枚搭載される。そして、一枚の IFC(InterFace Card) 対し、複数の FEC がバス接続される。FEC から IFC まではフローティングされ、外部の回路とは電気的に切り離されている。そのため、FEC から IFC までの回路の GND を、DSSD のストリップの電位と同じにすることができ、n 側ストリップでも DC 結合により読み出すことが可能である。

FEC-IFC BUS を流れる信号は、VA32TA のアナログ出力 (差動電流信号) と、トリガや各制御信号のディジタル信号である。ADC は IFC に搭載され、IFC 以降で外部の回路とやりとりされる信号は全てディジタル信号である。信号のやりとりは、LED と光センサなどを用いて電気的に絶縁して信号を伝達するデバイスである、ISOLATOR を通して行なわれる。ISOLATOR 以降の回路は DSSD のストリップと同じ電位である必要はなくなるため、他のディジタル機器 (VME BUS など) と共通の GND が使われる。IFC から出力されたデータは I/O 基板を通して、DP(Data Processer) に送られる。DP では送られてきたデータを処理し、最終的には VME BUS を通して PC などへ送られる。

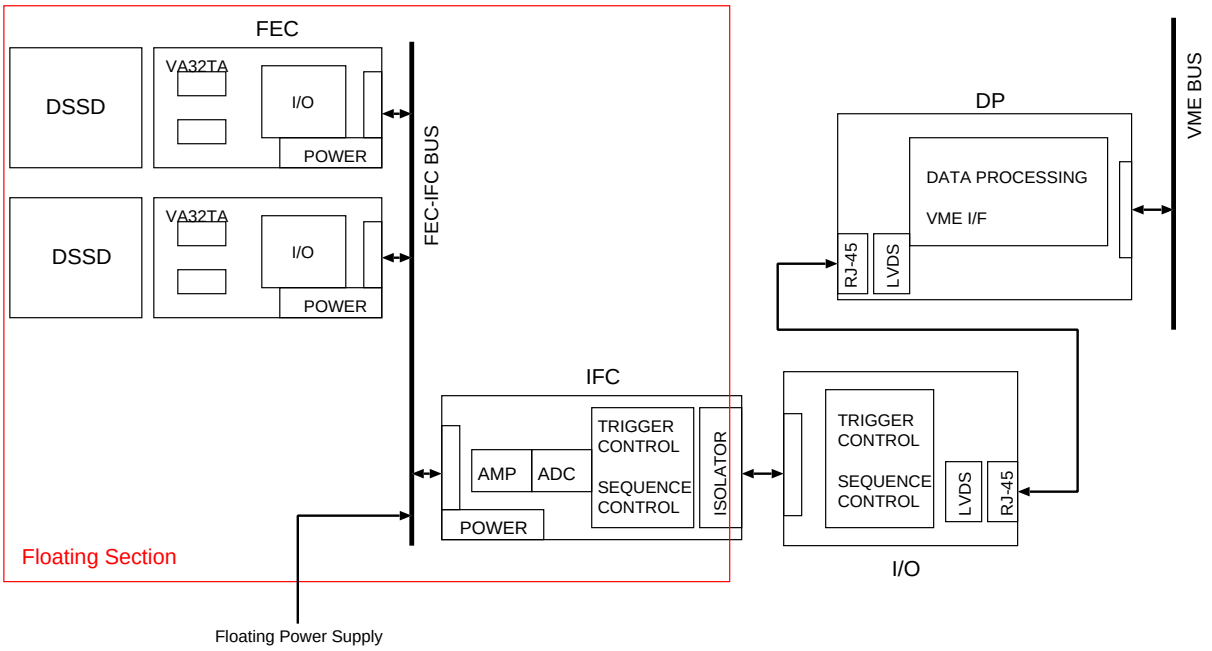


図 3.14: 新しい読み出しシステムのブロック図

新しい読み出しシステムは衛星搭載を考慮して全体的に小型化されており、フローティングに対応することで DSSD のバイアス周りを簡素化し、衛星に搭載されるコンプトンカメラの省スペース化、高信頼化も実現している。そして、FEC から I/O までが外部の回路から電気的に絶縁されることになり、フローティングに対応できること以外にも、VA-DAQ のような外部の機器によるノイズなどの影響を受けにくくなっているものと考えられる。また、FEC をバス接続するのに際して、各 FEC に電源レギュレータを設けている。そのため、FEC の枚数が多くなっても VA32TA に安定した電源を供給することができる。

さらに、新しい読み出しシステムは機能ごとにモジュール化されており、将来仕様が変更された場合でも、最小限の変更で対応することが可能である。現在のところ、この読み出しシステムは、VME BUS を使って制御しなければならない。しかし、I/O を SpaceWire(後述) に対応した物に変更することで、SpaceWire を通して制御・読み出しをすることが可能になる。さらに、FEC を交換するだけで VA32TA 以外のチップに対応させることもできるなど、高い拡張性を有している。また、IFC、FEC、I/O、DP には、それぞれ FPGA(後述) などのプログラマブルなロジックデバイスが搭載され、各モジュールがある程度独立して読み出しや VA32TA へのパラメータ書き込みなどの動作を行なう。そのため、VA-DAQ と比べてより柔軟な読み出しに対応しており、トリガーの立ったチップだけ読み出したり、チップごとに細かくパラメータを変えたりすることが可能になる。そして FPGA を書き換えるだけで仕様の変更にも容易に対応することができる。

第4章　SpaceWire

科学衛星では観測装置をはじめ、姿勢制御装置、通信装置、各種センサ、データ処理装置など、様々な機器の間で情報をやりとりする必要がある。現在、衛星上の装置間通信のためのハードウェア開発は、それぞれの装置の機能設計と同時に行われることが多い。このような設計手法では、各装置の機能を実現するのに最適なハードウェアを作ることができる反面、時間がかかり、コストも増大する。さらに、通信のためのハードウェアとそれぞれの装置の機能を実現するためのハードウェアが互いに影響を及ぼし合い、信頼性の低下にもつながる。また、衛星ごとに設計をやり直すため、過去の資産も継承されない。このような問題を打開すべく、従来より衛星上の装置間通信規格の共通化が考えられてきた。

衛星上の装置間通信の統一規格として、現在もっとも有力なものがSpaceWireである。SpaceWireはもともと、欧州宇宙機関 (ESA) によりIEEE1355をベースに宇宙用の標準規格として提案された。現在、SpaceWireの規格策定には、ヨーロッパをはじめ世界中の宇宙機関や企業が参加しており、日本からもJAXA/ISASや大阪大学を中心に参加している。

4.1　SpaceWireの特徴

SpaceWireは高速シリアル通信規格の一種であるが、データ転送レートは2〜400 Mbpsと可変であるため、様々な機器に柔軟に対応することが可能である。SpaceWireの信号伝送ではLVDS(Low Voltage Differential Signaling) と呼ばれる伝送方式が用いられている。低電圧の差動信号を用いて伝送するため高速通信でも低消費電力でノイズに強く、信号線からのノイズの輻射も少ない。また、Data-Strobe(DS) encodingと呼ばれる方式を採用しているため、Data信号とStrobe信号を伝送すれば、Data信号の読み取りのタイミングを合わせるのに必要なClock信号を受信側で再現することが可能である。図4.1に示すように、DataとStrobeのXORをとることでClockを再現できる。

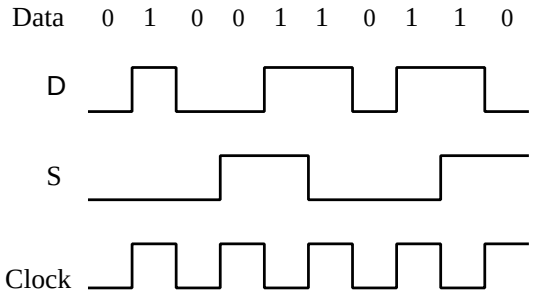


図 4.1: Data-Strobe(DS) encoding

SpaceWire に用いるケーブルは、Data、Strobe が入出力合わせて 2 組で構成されている。それぞれの信号線は、2 本のケーブルを強くよりあわせたツイストペアになっており、ノイズに強い構造となっている。ケーブルは最大で 10 m 以上延ばすことができるため、衛星上での自由な機器配置が可能である。また、使用されるコネクタは 9 ピンの D-sub コネクタと同じ形状である。

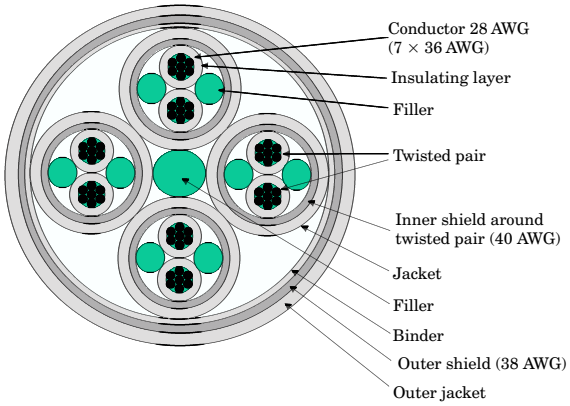


図 4.2: ケーブルの構造 [5]

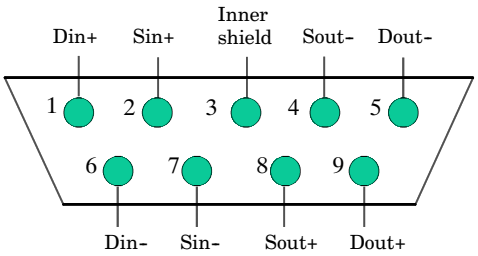


図 4.3: コネクタ・ピンアサイン [5]

SpaceWire 上を流れるデータは、ある長さに分割されたデータにあて先情報を付加したパケットとしてやりとりされる。SpaceWire はネットワークに対応しており、ルータと呼ばれる機器を組み合わせることにより、メッシュ状のネットワークを構成することもできる。また、ネットワーク上のある回線が切れた場合でも、迂回ルートでデータを送ることができるため、信頼性の高いシステムを構築することができる。さらに、転送レートが足りない場合は、複数の回線を同時に使用することで、より高速なデータ転送も可能にしている。

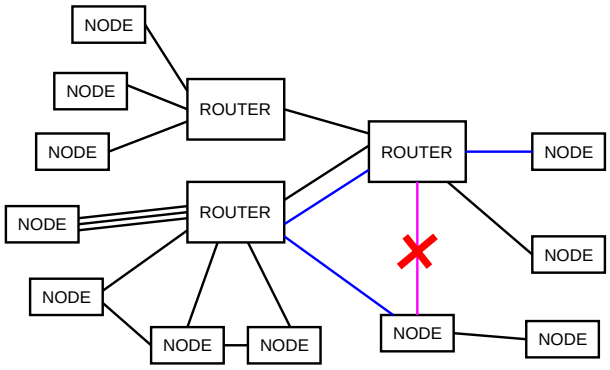


図 4.4: SpaceWire のネットワーク

SpaceWire で接続された装置の情報には、リモートメモリアクセスと呼ばれる手法により、メモリマップド I/O でアクセスすることが可能である。コンピュータの CPU がメモリ上のデータを読み書きする場合、アドレスによりメモリ上の場所を指定している。CPU が周辺機器と接続される場合、CPU が I/O 専用のレジスタを持つ場合もあるが、メモリと同じアドレス空間上に周辺機器のレジスタを配置し、メモリにアクセスするのと同じ手順で周辺機器の情報にアクセスする方法をメモリマップド I/O と呼ぶ。SpaceWire に接続された機器は自分のバスに接続された機器のようにメモリマップド I/O によるアクセスが可能となる。この機能の実現には、簡便なプロトコルチップを用いれば良く、CPU を持たない機器も SpaceWire のネットワークに接続することができる。

以上のように、SpaceWire は様々な機器に柔軟に対応でき、信頼性も高い。SpaceWire を用いた科学衛星開発では、通信インターフェイスの開発とそれぞれの機器開発を完全に分離することができ、各機器は互いに干渉すること無く、容易に相互接続することができるようになる。さらに、プロトコルが簡便なことから、必ずしも専用のデバイスを用意する必要は無く、FPGA と LVDS のインターフェイスさえあればよい。そのため、非常に低コストで実現することができる。このような特徴から、車載機器や通信機器業界からの関心も高く、民生機器への応用も期待される。

4.2 SpaceCube と FADC ボード

4.2.1 SpaceCube

SpaceCube は、SpaceWire の開発プラットホームとして開発された超小型 PC で、JAXA/ISAS とシマフジ電機により開発された。CPU には MIPS アーキテクチャを持つ VR5701(300 MHz) を搭載し、一般的な PC の機能に加えて SpaceWire を 3 ポート搭載している。Linux と T-Engine が移植されており、キーボード、マウス、ディスプレイなどを接続すると、普通の PC としても使用できる。SpaceCube は民生機器の技術を多用しており、非常に低コストで SpaceWire のテスト環境を実現している。

Linux の場合は全てのプロセスが OS によって厳密に管理されているため、例えば割り込みが入った場合でも、OS の影響によりプロセスに割り込みが伝わるタイミングにずれが生じる場合がある。そこで、OS として Linux よりもリアルタイム性に優れた T-Kernel を使用した方が、高度なタイミング制御が必要な組み込み機器としては有利である。しかし Linux は現在 PC で広く使われており、使い慣れた Linux 環境上で開発ができるというメリットがある。

CPU	VR5701 200 MHz / 250 MHz / 300 MHz
Flash ROM	16 MB
DRAM	DDR SDRAM 64 MB
INNPUT OUTPUT	IEEE1355(SpaceWire),RTC,CF(True IDE),XGA(1024×768),USB1.1, LAN(100BASE),Audio(Stereo), RS232C,JTAG I/F(Debug)
POWER	+5 V
SIZE	52 mm × 52 mm × 55 mm

表 4.1: SpaceCube の仕様

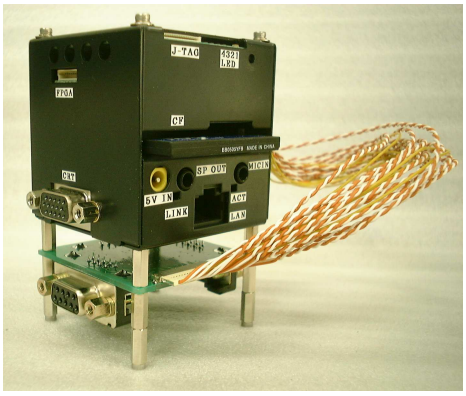


図 4.5: SpaceCube の写真

4.2.2 FADC ボード

FADC ボードは、SpaceWire を搭載したフラッシュADC ボードである。4 つのアナログ入力を持ち、搭載されている ADC(ANALOG DEVICES AD9240) は 14 bit、 最高 10 MSPS で動作する。また、ディジタル入力と出力をそれぞれ 2 個づつ持っている。そして、2 個の FPGA(ALTERA 社 Cyclone シリーズ EP1C6T144C6N) を搭載し、一方の FPGA には SpaceWire の機能が実装されている。もう一方の FPGA は User FPGA と呼ばれ、ユーザが独自の回路を書き込むことができる。User FPGA の回路を書き換えることにより、ADC で取り込んだデータに目的の処理を施してから SpaceWire で取り込むことが可能である。

現在、大阪大学の能町教授により、FPGA の回路および SpaceCube の Linux 上で動作する SpaceWire 関連のソフトウェアが開発されており、入力されたアナログ信号の波形を SpaceWire のリモートメモリアクセス機能を使って取得できるようになっている。今回、この FADC ボードと SpaceCube を用い、実際に SpaceWire によるデータのやりとりを行い、さらに User FPGA に独自の回路を書き込むことを試みた。図 4.6 に FADC ボードの写真を、図 4.7 に FADC ボードのブロック図を示す。

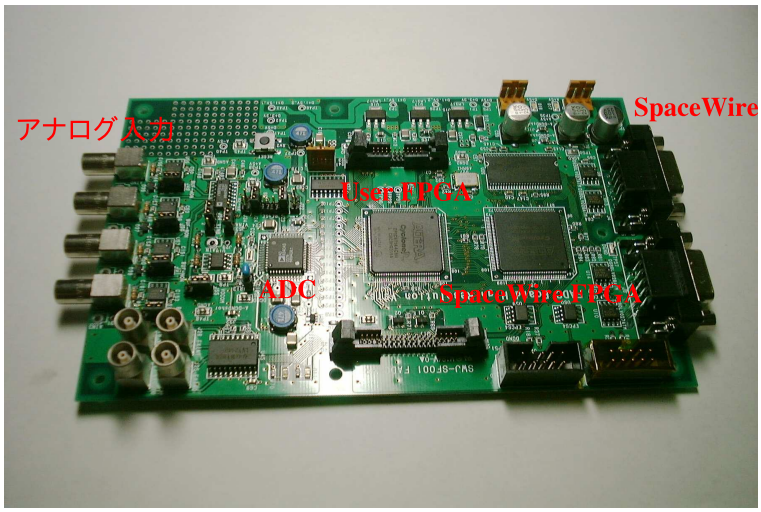


図 4.6: FADC ボードの写真

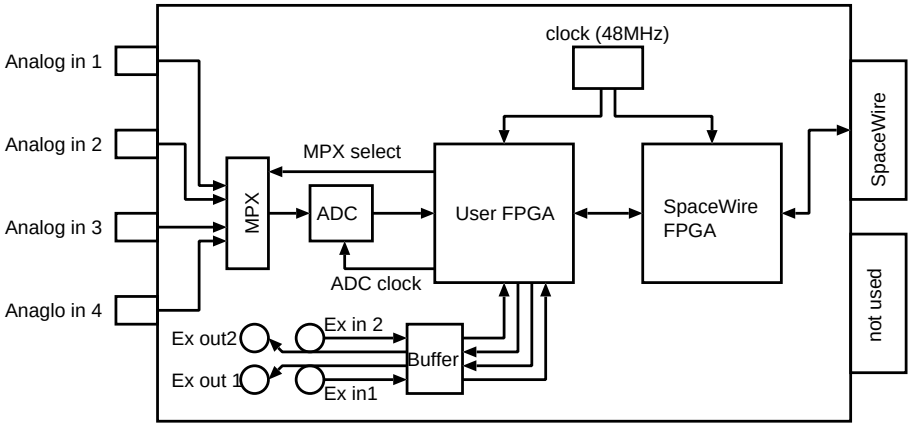


図 4.7: FADC ボードのブロック図

4.2.3 FPGA

通常、ディジタル信号を処理するための回路は、標準ロジック IC などと呼ばれる IC を多数並べて作る。しかし、実装面積が大きくなり、高速な動作が難しいといった問題がある。そこで、特定用途向けの集積回路である ASIC(Application Specific Integrated Circuit) が使われる。ASIC を作ることで、回路を一つの集積回路に納めてしまい、省面積化、高速化を実現することができる。しかし、このような ASIC は半導体工場で製造されるため、開発に時間がかかり、工場の設備を使用するために莫大なコストもかかる。そのため、ASIC の開発段階での試作や、少量の ASIC しか使用しない場合、その度に半導体工場で LSI を生産するのは非現実的である。また、製造された LSI の中身は書き換えることができないため、後から回路を修正したりすることは不可能である。そこで、ユーザが自由に回路を書き換えることのできるデバイスとして開発されたのが FPGA(Field

Programmable Gate Array) である。同じように回路を書き換えることのできるデバイスとして CPLD(Complex Programmable Logic Device) と呼ばれるものもあるが、両者では内部構造や書き込める回路規模に違いがあり、一般に FPGA の方がより大規模な回路を書き込むことができる。初期の頃の FPGA は性能が悪く、非常に高価で、ASIC 開発時の試作や研究にのみ用いられていたが、現在では性能も上がり価格も下がってきたことから、試作以外にも実際の製品に組み込まれて使用されることも多く、人工衛星でもすでに使われている。

ここでは、ALTERA 社の FPGA である Cyclone シリーズを例に説明するが、他の FPGA でも基本的に類似の構造である。FPGA 内部はロジック・アレイと呼ばれる論理ブロックが多数敷き詰められた構造をしており、これがユーザの論理を実現する。図 4.8 に、Cyclone シリーズのロジック・アレイのブロック図を示す。ロジック・アレイは、ロジック・アレイ・ブロック (LAB) と LAB

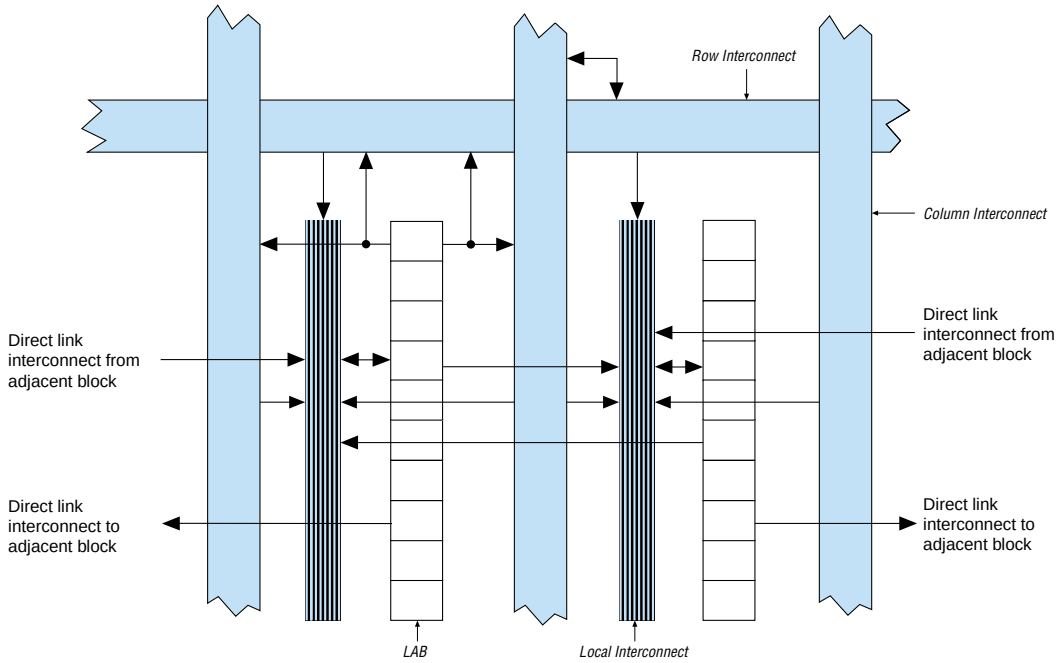


図 4.8: ロジック・アレイのブロック図 [1]

同士をつなぐ配線により構成されている。LAB はさらに 10 個のロジック・エレメント (LE) により構成されており、LE がもっとも基本的な論理ブロックとなっている。図 4.9 にロジック・エレメントのブロック図を示す。1 つの LAB 内の LE はローカル配線 (Local Interconnect) により接続され、LAB 同士はロウ配線 (Row Interconnect) とカラム配線 (Column Interconnect) により接続される。また、横方向に隣り合った LAB 同士はダイレクト・リンク配線 (Direct link interconnect) により接続することができる。これらの配線を用いて LE を組合わせることにより、様々な論理を実現しユーザ独自の回路を組み込むことができる。FPGA には数千から数万個の LE が搭載されており、LE の数が書き込める回路の規模を決めている。FPGA にはこのような論理ブロックに加

えて、外部の回路と接続するためのI/Oが搭載されているが、FPGAによってはPLLやRAMが搭載されているものもあり、FADCボードに使用されているFPGAでは2個のPLLと92160 bitのRAMを搭載している。

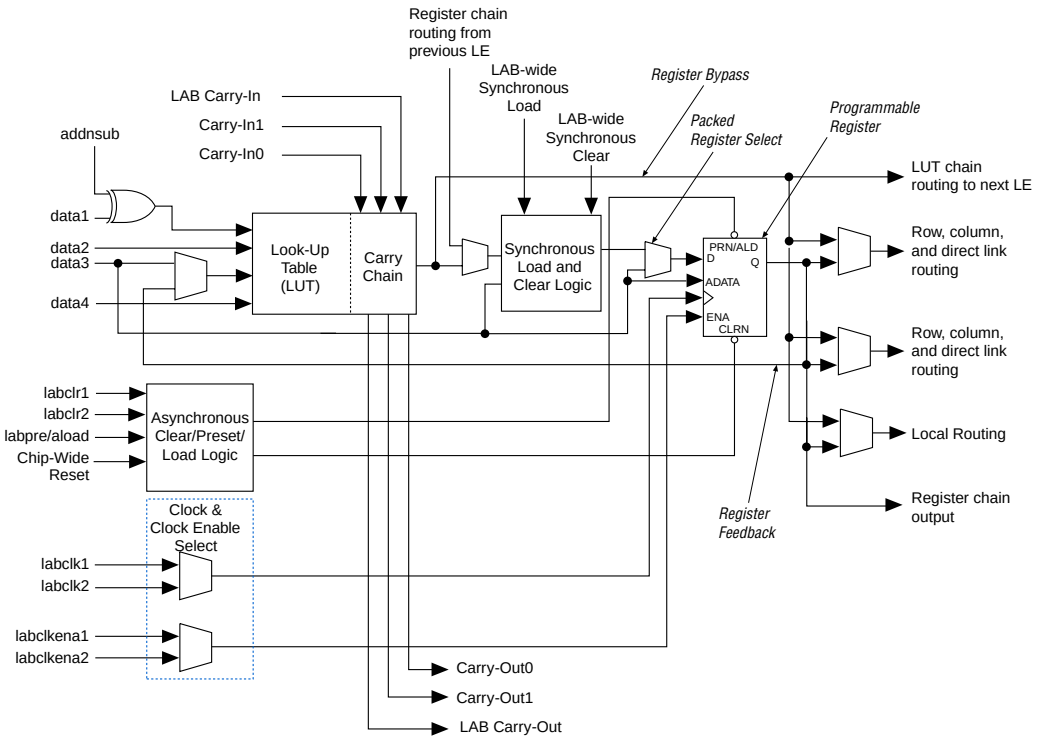


図 4.9: ロジック・エレメントのブロック図 [1]

一般的なFPGAの場合、書き込まれた回路の情報はSRAM(Static Random Access Memory)のアーキテクチャにより保持される。SRAMは電源を切ると情報を失ってしまうといった特徴があるため、FPGAは電源を入れ直すたびに回路情報を書き込む必要がある。そこで、外部に回路情報を記憶したROM(コンフィグレーションROM)を実装し、電源投入時に自動的に回路情報を読み込むようにして使われることが多い。FADCボード上のFPGAもコンフィグレーションROMから回路情報を読み込むようになっている。そのため、FADCボード上のFPGAの回路を書き換えるには、コンフィグレーションROMを書き換える必要がある。Cycloneシリーズに用いられるコンフィグレーションROMはEEPROM(Electronically Erasable and Programmable Read Only Memory)と呼ばれる電気的に書き換え可能なROMで、10000回以上の書き換えに対応している。

4.2.4 VHDL

従来、ASICやFPGAなどの設計では、ANDやORなどのロジックゲートを組み合わせた回路図を描いて目的の動作を記述する、回路図入力が用いられてきた。しかし、回路図入力では手間がかかるうえ、目的の動作を実現するために複雑な論理式を考える必要もあった。そのため、設

計期間が長くかかり、途中からの回路変更も容易では無かった。そこで、回路の動作をコンピュータのプログラム言語のように記述できる、ハードウェア記述言語 (HDL: Hardware Description Language) が考案された。HDL を用いることにより、回路の動作を機能で表現することができ、複雑な論理式を考えなくても回路を設計することが可能となった。その結果、開発期間を大幅に短縮でき、設計の変更も容易に行えるようになった。

現在最も普及している HDL には VHDL と Verilog HDL があるが、本研究では VHDL(Very High Speed Integrated Circuit HDL) を用いた。元々VHDL は米国国防省によって提唱されたものであるが、その後米国電気電子技術者協会 (IEEE) により標準化され、現在では世界に広く普及している。VHDL は記述能力が高く、システム全体のアーキテクチャからロジック・ゲートレベルの記述にまで対応している。VHDL を実際に ASIC や FPGA の回路を記述するのに用いる場合は、RTL(Register Transfer Level) と呼ばれるレベルで記述する。RTL では、CAD ツールにより VHDL の記述から自動的にロジック回路を生成 (論理合成) することができる。また、RTL より抽象度の高い記述レベルは、システム全体のアーキテクチャのシミュレーションや、アルゴリズムの検証などに用いられる。現在、より抽象度の高い記述から直接回路を合成する研究も行なわれているが、まだ一般的ではない。以下に VHDL によるロジック・ゲートの記述の例を示す。また、図 4.10 に論理合成の結果を示す。

```
library IEEE;
use IEEE.std_logic_1164.all;

entity gates is
        port (  A : in std_logic;
                B : in std_logic;
               Y1 : out std_logic;
               Y2 : out std_logic;
               Y3 : out std_logic;
               Y4 : out std_logic);
end gates;

architecture RTL of gates is
begin

        Y1 <= A and B;
        Y2 <= A or B;
        Y3 <= A nand B;
        y4 <= A xor B;

end RTL;
```

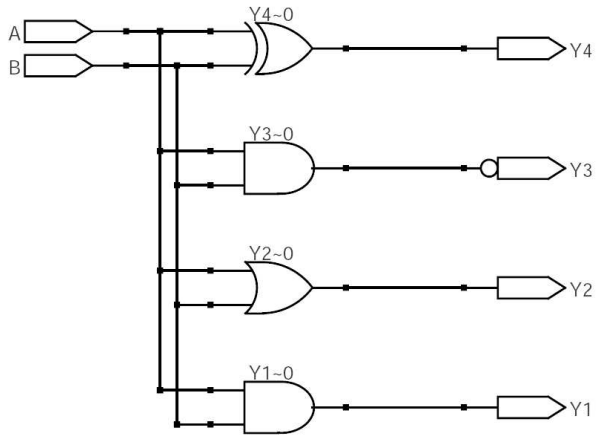


図 4.10: VHDL でロジック・ゲートを記述した例

前述のように、VHDL での記述を実際の FPGA の回路設計に用いる場合、論理合成をする必要がある。現在では、多くの FPGA メーカが自社のデバイスに対応した開発環境を無償で提供しており、開発環境に論理合成ツールが含まれている。また、シミュレータを用いれば設計した回路が正しく動作するかを事前に検証することもできる。本研究では ALTERA 社の開発環境である Quartus II Web Edition Software Version 5.1 を用いた。

4.3 SpaceCube と FADC ボードを用いたマルチチャンネルアナライザの製作

SpaceWire を使ったデータのやりとりおよび、FPGA や VHDL の扱いに慣れるため、SpaceCube と FADC ボードを用いてマルチチャンネルアナライザ (MCA) を製作した。そして、設計通りの機能を実現できているかの評価を行なった。

4.3.1 SpaceCube の立ち上げ

SpaceCube は入手したままの状態では OS がインストールされておらず使えないので、Linux をインストールする必要がある。SpaceCube は HDD を持たず、代わりに CF(Compact Flash) を使用する。そのため、別途 1 GB の CF を用意し、ここに Linux のファイルシステムを作成した。また、Linux の中核である Kernel は SpaceCube 内蔵の FlashROM に書き込み、そこから起動する。SpaceCube に付属している Linux は、Red Hat Linux 7.1 である。

SpaceCube にディスプレイやマウス、キーボードなどを接続すれば普通の PC のように使うこともできるが、今回これらは不要であるため、シリアルコンソールのみで操作した。シリアルコンソールで操作するには、SpaceCube のシリアルポートと PC を接続し、PC 上のコンソールアプリケーションを使用する。Linux のインストールが完了するまではシリアルコンソールからしか操作できないが、インストール完了後は SpaceCube と PC をネットワークケーブルで接続

し、SpaceCube に telnet や SSH で PC から login して操作することができるようになる。また、SpaceCube と FADC ボードを接続する SpaceWire のケーブルは付属していなかったため、D-Sub コネクタと単芯のケーブルを寄り合わせてツイストペアにしたもので代用した。短距離の接続であるので、このようなケーブルでも問題なく通信できている。

以上のようにして SpaceCube と FADC ボードを使える状態にした。そして、能町教授が作成されたプログラムを用いて、正常に SpaceWire を通して FADC ボードにアクセスできることが確認された。

4.3.2 MCA の開発 (ハードウェア)

SpaceCube、FADC ボードとも正常に動作することが確認できたため、FADC ボードを使った MCA を開発した。MCA とは、入力されたアナログパルス信号のピーク値を AD 変換し、その度数分布を得るための装置で、放射線のスペクトルを得るために用いられる。まず、製作する MCA の仕様を以下のように決めた。

- ピークホールドされた信号を 2ch 入力する
- トリガも外部から 2ch 入力する
- どのチャンネルのトリガが立ったかを記録する
- イベントごとに 1 ms 間隔程度の時刻情報を付加
- 最高 10 kHz までのイベントレートに対応

また、FADC ボードと SpaceCube、周辺装置類は図 4.11 のブロック図のように接続した。

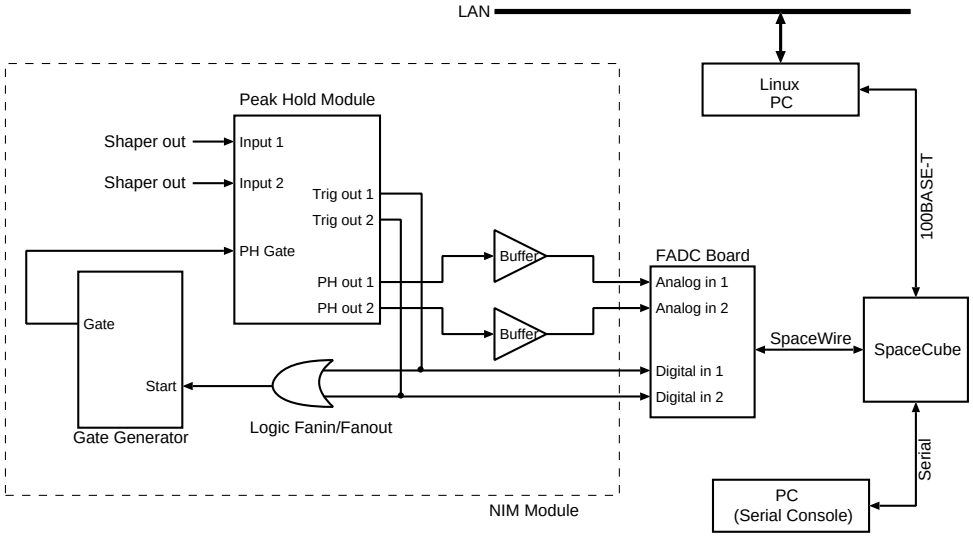


図 4.11: セットアップ全体のブロック図

FADC ボードにはピークホールドの機能は無いため、外部のピークホールドモジュール (CLEAR PULSE PIN-LSI TEST BOARD CP-2835) を使用した。ピークホールドモジュールには Discriminator も内蔵されており、トリガもピークホールドモジュールから出力される。トリガ出力は 5V の TTL であるのに対し、FADC ボードは 3.3V 系で構成されているが、FADC ボードのディジタル入力端子に接続されているバッファに 5V トレラント機能があるため、そのまま接続しても問題ない。また、ピークホールド出力の電流ドライブ能力が高くなく、FADC ボードの入力インピーダンス (50 Ω) にマッチしないため、バッファIC(BURR-BROWN BUF634) を用いたバッファアンプを製作し、ピークホールドモジュールと FADC ボードの間に設けた。ピークホールドモジュールは Peak Hold Gate 信号が入力されている間、入力されたアナログ信号のピーク値を保持するものであるが、ピークホールドモジュールから出力されたトリガ信号はそのままではパルス幅が短過ぎ、Peak Hold Gate 信号として使うことができないため、Gate Generator で Peak Hold Gate 信号を生成する。Gate Generator は、トリガ信号を元に任意のパルス幅を持った信号を生成するモジュールである。図 4.12 にピークホールド信号生成のタイミングチャートを示し、図 4.13 に、FADC ボードに入力する信号を生成するための装置の写真を示す。装置は全て NIM モジュールで構成されている。モジュールは左から、テスト信号を生成するための Pulser、Shaping Amp、トリガを数えるための Visual Scaler(デバック用)、トリガ信号の OR をとるための Logic Fanin/Fanout、ピークホールドモジュールのロジックレベル (TTL) とその他の機器のロジックレベル (NIM) を変換するための Logic Level Adapter、Gate Generator、ピークホールドモジュール、製作したバッファアンプである。また、図 4.14 に SpaceCube と FADC の写真を、図 4.15 にセットアップの全体写真を示す。

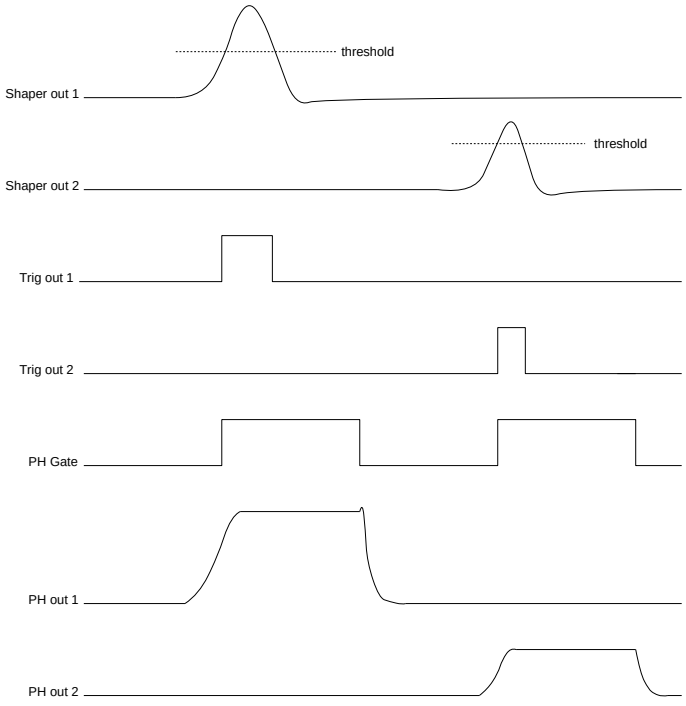


図 4.12: ピークホールド信号生成のタイミングチャート

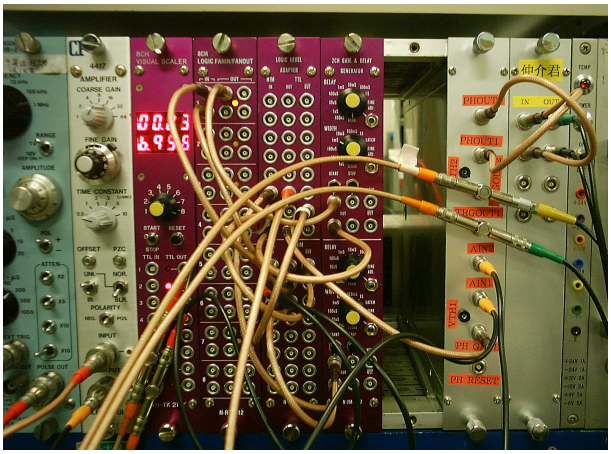


図 4.13: NIM モジュール

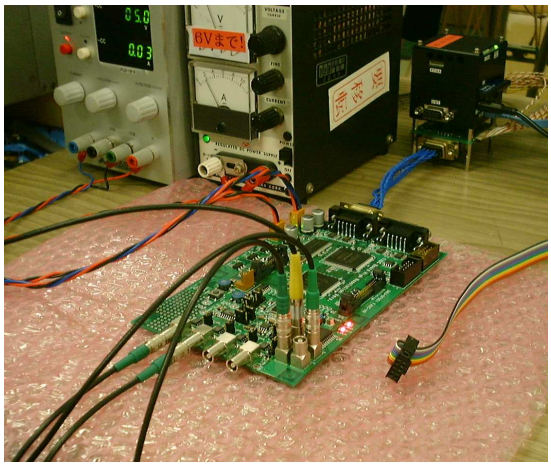


図 4.14: FADC ボードと SpaceCube

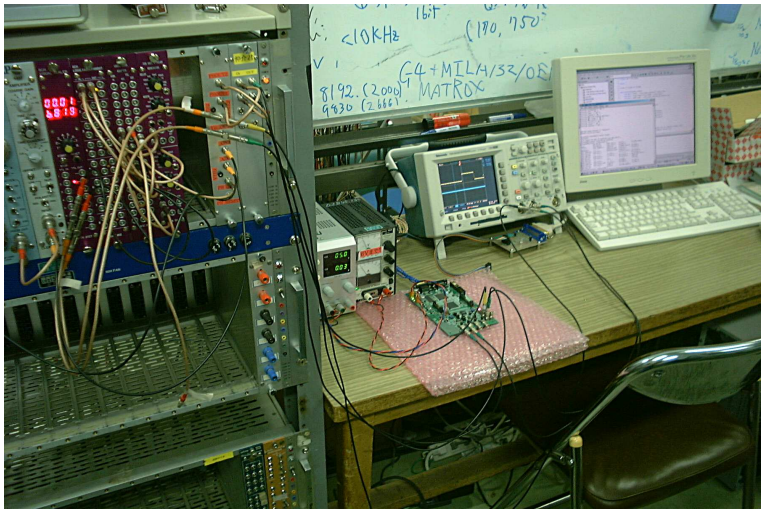


図 4.15: セットアップの全体

ここまでがアナログ信号の入力から SpaceCube までの構成であるが、FADC ボードから SpaceCube に読み込まれたデータは、SpaceCube 上に置いたままでは解析などの作業ができず、CF の容量も限られているため、実験室レベルでは、得られたデータをすぐに PC に移せた方が便利である。そこで、シリアルコンソールとは別に、SpaceCube の LAN ポートを PC と接続し、SpaceCube 上のデータを外部に出せるようにしている。SpaceCube と接続する PC には NIC(Network Interface Card) を 2 枚搭載し、一方は SpaceCube に、もう一方は研究室内の LAN に接続している。このようにすることで、SpaceWire を通して得られたデータをスムーズに解析用の PC などに転送できる。

以上のようなセットアップと仕様に対応するため、図 4.16 のようなブロック図の機能を持った回路を VHDL で記述し、User FPGA に実装した。この内、SpaceWire FPGA との通信部分は能

町教授が作成された回路をそのまま使用している。回路の動作クロックには 48 MHz を 8 分周した 6 MHz を用い、それとは別に、時刻カウンタ用に 32768 分周した 1464.84375 Hz のクロックも生成している。これにより、682.67 μs 間隔での時刻情報を得ることができる。また、デッドタイムカウンタ用に 32 分周した 1.5 MHz のクロックも生成しており、667 ns 単位でデッドタイムを測定することができる。ここで、内部バスはデータバスが 16 bit 幅、アドレスバスが 10 bit 幅である。10 bit のアドレスを UserFPGA 内部のレジスタに割り当てておくことで、アドレスを割り当てたレジスタには SpaceWire からのリモートメモリアクセスが可能になる。そして、外部にディジタル信号を出力できる端子が 2 個あるため、一方に外部からデッドタイムを監視するための busy 信号を、もう一方にはデバック用に 2 つのトリガ信号の OR をとったものを出力している。

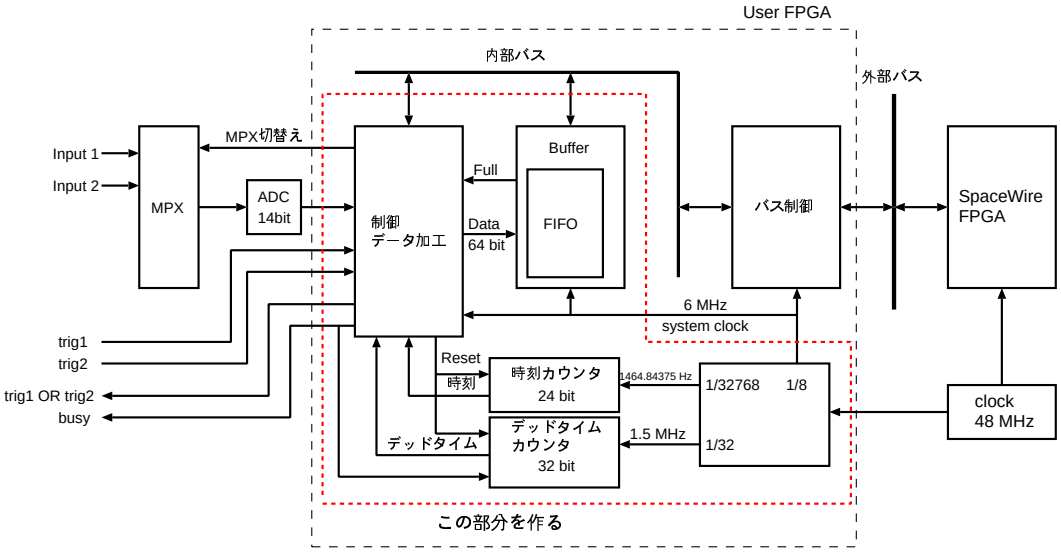


図 4.16: MCA のブロック図

時刻カウンタ・デッドタイムカウンタ

時刻カウンタは、イベントが起こった時刻を記録するためのもので、リセット入力とクロック入力、時刻出力を持ち、クロックごとに 1 つずつ値が増えて行く 24bit のカウンタである。測定開始時にリセットすることにより、測定を開始してからの経過時間としてイベントの時刻情報を得ることができる。時刻カウンタに供給するクロックは 1464.84375 Hz であり、682.67 μs に 1 回カウントアップする。24 bit のカウンタであるため、最高 11453.25 秒 (3.2 時間) までの時刻を記録することができる。3.2 時間を越える測定を行なうことも考えられるが、時刻の情報は 1 イベントごとに記録しているため、3.2 時間の間に 1 度もイベントが無かった場合を除けばカウンタが 1 周したことを後から知ることができるので実用上は問題ない。

デッドタイムカウンタは、MCA がデータを取ることのできない時間であるデッドタイムを積算するためのカウンタで、時刻カウンタとほぼ同様のものである。ただし、デッドタイムを計るために busy 信号入力を持っている。busy 信号は MCA がデータを取ることのできない間出される信

号で、busy 信号とクロック信号の AND をとったものでカウンタを駆動することで、busy 信号が立っている間だけカウンタはクロックに合わせてカウントアップする。デッドタイムカウンタに入力するクロックは 1.5 MHz であり、32 bit のカウンタであることから、最高で 2863.312 秒 (47.7 分) までのデッドタイムを記録することができる。時刻カウンタとリセット信号を共有しており、測定の最後にデッドタイムカウンタの情報を読み取ることで、その計測時間に占めるデッドタイムの割合を知ることができる。

制御・データ加工

制御・データ加工を行なう回路ブロックでは、トリガの検出、マルチプレクサの切替え、ADC 出力の取り込み、時刻情報の付加、バッファへのデータ転送を行なっている。また、デッドタイムカウンタの値を読み出すためのレジスタも含まれている。まず、図 4.17 にトリガとピークホールド出力のタイミングを示す。図 4.17 のように、トリガが立ってからピークホールド信号が安定す

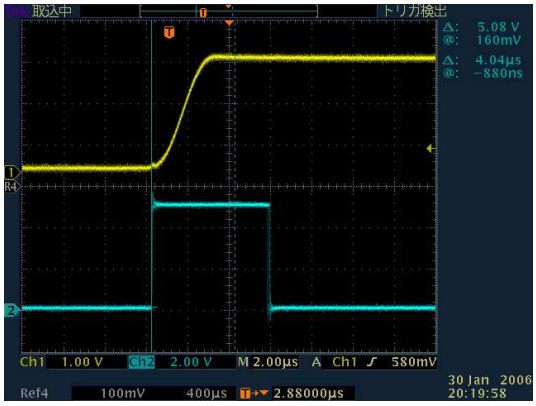


図 4.17: トリガとピークホールド出力のタイミング
(上の波形がピークホールド出力、下がトリガ)

るまでに時間がかかる。この時間はピークホールドモジュールに入力された信号の大きさや波形整形の時定数でも変化し、大きな信号や長い時定数ほど時間がかかる。オシロスコープで実測したところ、今回のセットアップでは安定するまで 4μs 程度かかっていた。そこで、トリガが立ってからピークホールド信号が十分安定する 5μs 以上経過してから最初のチャンネルの AD 変換値を取得する。また、ch1 の AD 変換が終了したらマルチプレクサを切替え、ch2 の AD 変換をするが、マルチプレクサが切り替わってからマルチプレクサ出力が安定するまで数 100 ns の時間がかかるため、ここでも信号が安定するまで待機する必要がある。また、ADC でサンプリングされたデータが実際に出力されるまで 3 clock の遅れがある。以上のことを考慮して、AD 変換は以下の手順で行なう。

1. どちらかのチャンネルでトリガが立つのを待つ
2. ch1 か ch2 でトリガが立ったら時刻情報を保持して 5μs 以上待つ

3. トリガが立ったチャンネルの情報 (どのチャンネルでトリガが立ったか) を保持して ch1 の AD 変換値を保持する

4. マルチプレクサを ch2 に切替えて数 100 ns 待つ

5. ch2 の AD 変換値を保持する

6. バッファが溢れていなければ得られたデータをバッファに書き込む

7. マルチプレクサを ch1 に切替えて数 100 ns 待つ

8. トリガが Low になっていたら 1. へ戻る

以上の手順を回路にすれば良いが、このようなシーケンス処理をディジタル回路で行なう場合、ステートマシンが用いられる。ステートマシンとは、過去の出力結果と、現在の入力の状態、そして決められた手順によってその出力が決まる回路である。つまり、ある条件が成り立ったとき、ステートマシンは次の状態に遷移し、遷移すると同時に決められた処理を行なう。このような状態の遷移や各処理はクロック信号に合わせて行なわれる。一般にステートマシンは非常に複雑な回路となり、ディスクリート回路で構成するのは容易ではないが、VHDL では容易にステートマシンを記述することができる。前述の手順をステートマシンにしたところ、以下のようなものとなった。図 4.18 に作成した部分の詳細なブロック図を、図 4.19 にステートマシンの状態遷移図を示す。

ここで注意すべきことは、トリガ入力をフリップフロップで受けていることである。トリガ入力はクロックと関係なく変化する非同期信号であるため、同期信号しか入力されないことを前提にしている回路に非同期信号が入ると、クロックとのタイミング次第で誤動作を起こす。信号をフリップフロップで受けている場合、フリップフロップからの出力はクロックに同期したものとなるため、誤動作を起こすことは無い。

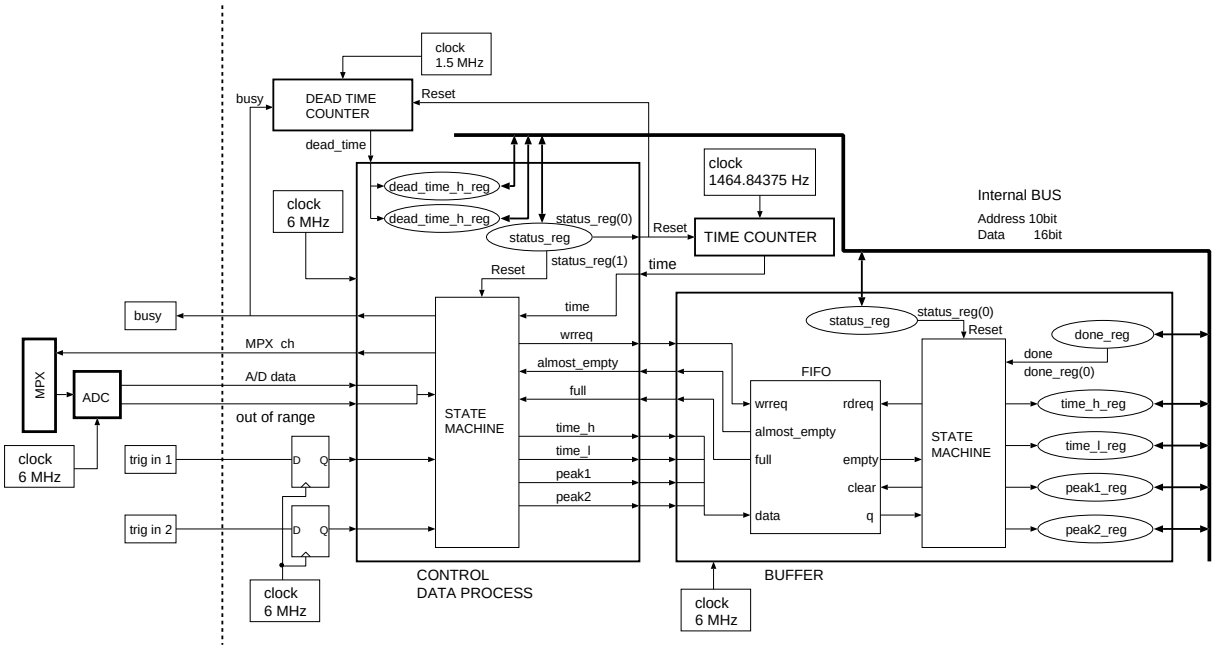


図 4.18: 詳細なブロック図

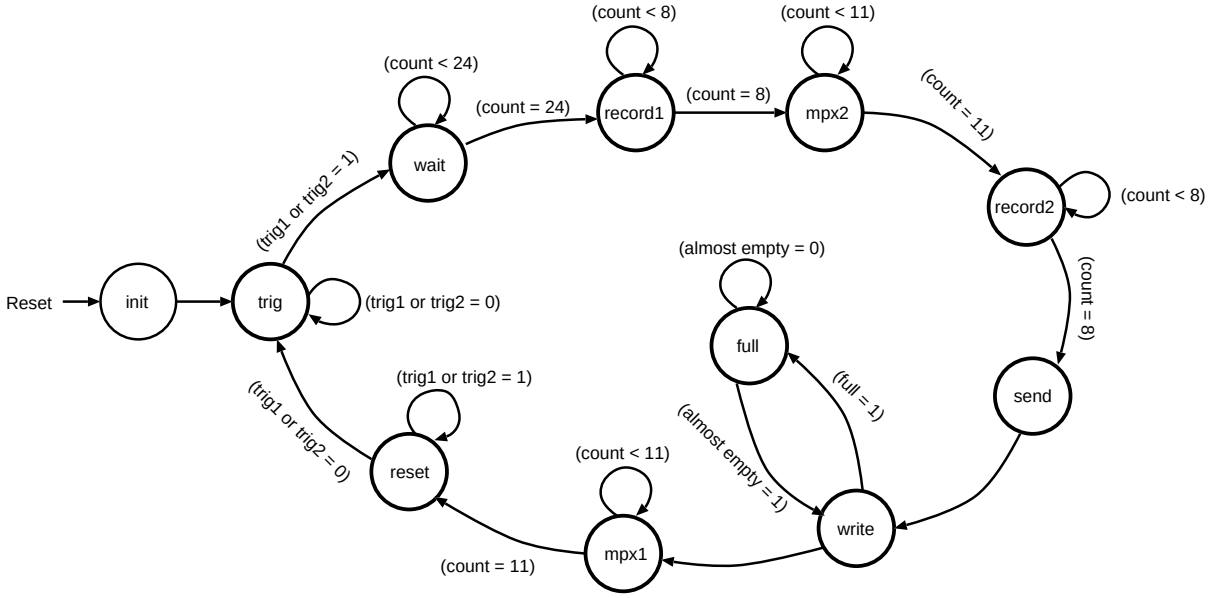


図 4.19: データ取得の状態遷移図

状態 trig

まず、リセットがかかった後、トリガの入力を待つ状態 trig に入る。この間、2 つのチャンネルのトリガの OR をとった信号を監視し、トリガが立っていない限りこの状態のままでいる。そして、どちらかのチャンネルでトリガが立った場合、次の状態 wait に遷移する。その際、時刻カウ

ンタからのデータを時刻保持用のレジスタに代入し、トリガが立った時刻を記録する。また、トリガが立ったチャンネルの情報を、いったん別のレジスタに保持しておく。この状態以外の状態にいるとき、データの取得はできずにデッドタイムとなる。そこで、状態 trig 以外の状態のときに busy 信号を出力するようにして、デッドタイムカウンタや外部からデッドタイムを測定できるようにした。

状態 wait

状態 wait では、ピークホールド出力が安定するまで待機する。そのためにカウンタを用意しておき、1 クロックに 1 回カウントアップするようにしておく。カウンタの値がある一定の値になるまでこの状態のままでいることで、ある一定時間待機することができる。この場合、クロックが 6 MHz で 5 μs 以上待機する必要があることから、カウンタが 36 になったら次の状態 record1 に遷移することにした。これにより、6 μs 待機することになる。さらに、待機している間、最初にトリガが立ったときに保持したトリガの情報と、現在のトリガの情報の OR を取り、それを最終的なトリガの情報とすることで、最初にトリガが立ってから 6 μs 以内にもう一方のトリガが立った場合は、2 つのチャンネルで同時にトリガが立ったものとみなす。

状態 record1

状態 record1 では、ch1 について AD 変換した値を記録する。ここではノイズの影響を低減し、AD 変換の精度を向上させるため、8 回サンプリングした値を平均したものを記録することにした。そのために、カウンタが 8 になるまで AD 変換した値を足し続け、後でバッファに転送する際に 8 で割る。ADC の出力は OTR 信号を入れても 15 bit であるが、16 bit 単位でデータを扱った方が都合が良いので、ADC 出力は 16 bit のレジスタに入っている。そして、16 bit のデータを 8 回足すと 19 bit のデータになるが、都合により足した値を保持するレジスタは 20 bit としている。8 回の足し算が終わると、次の状態 mpx2 に遷移する。

状態 mpx2

状態 mpx2 では、マルチプレクサの切替えを行なう。前の状態からこの状態に遷移する際にマルチプレクサの切替え信号が出力されるため、このステートではピークホールド信号が安定するまで待ったときと同様に、マルチプレクサの出力が安定するまで待機する。この場合、カウンタが 11 になるまで待機し、1.83 μs 待機するようにしている。これはマルチプレクサ出力が安定するには十分な時間である。

状態 record2

状態 record2 では、ch1 と同様に ADC の出力を 8 回足し算する。これで ch1 と ch2 の AD 変換値を得ることができた。

状態 send

状態 send では、これまでに得られている時刻、トリガパターン、AD 変換値を、バッファの回路ブロックに接続されているポートに出力する。内部のデータバスの幅が 16 bit であるため、最終的に 16 bit のデータを出力しなければならない。そのため、この時点で 24 bit の時刻情報は上位

8 bit と下位 16 bit に分割して出力する。そして、トリガパターンの情報は 2 bit あれば良いため、時刻情報の上位ビットに付加することにする。また、ch1 と ch2 の AD 変換値を 8 回足した値の入っている 2 つの 20 bit(19-0 bit) レジスタの、それぞれ 18 から 3 bit 目までを切り出して 16 bit のデータとすることで、3 回の右ビットシフトにより 8 で割ることになるので、ch1 と ch2 の AD 変換値の 8 回平均を求めることができる。このようにして、1 回のイベントで合わせて 64 bit のデータを得る。

状態 write
状態 write では、前の状態から遷移するときにポートに出力した値を実際にバッファに書き込む。ただし、バッファが溢れている場合はデータを書き込むことができないので、バッファから出力される full 信号を見て、バッファに空きがあることを確認する。空きがある場合には、書き込み要求する信号を出力し、状態 mpx1 に遷移する。もし full 信号が 1 でバッファに空きが無い場合は、状態 full に遷移する。

状態 full
状態 full ではバッファからの almost empty 信号が 1 になるまで待機する。almost empty 信号は、バッファの中のデータがほとんど空 (この場合は 1 個) になったときに出される信号で、バッファが完全に空になる前にバッファへの書き込みを再開し、読み込み側でバッファが空になって待機する時間が生じることを防ぐために出力される。almost empty 信号が 1 になると、状態 write に戻る。

状態 mpx1
状態 mpx1 では、マルチプレクサを再び ch1 に切替える。mpx2 でマルチプレクサを ch2 に切替えたのと同様であるが、最初に状態 write で出力した書き込み要求信号を取り下げる処理を行なう。

状態 reset
状態 reset では、両方のチャンネルのトリガ信号が Low になるまで待機し、一つのイベントで何度もデータを取ることを防ぐ。トリガが Low なら状態 trig に遷移する。

バッファ

イベントごとに得られたデータは、いったんバッファに格納されてから SpaceCube で読み出される。イベントが起こる度にレジスタの値を読んでも良いのだが、SpaceWire によるデータ転送や SpaceCube 側での処理にある程度時間がかかるため、その間は次のイベントが起こってもデータを取ることができない。このデータを取ることのできない時間はデッドタイムと呼ばれ、イベントレートに対してデッドタイムが十分短いことが求められる。実際の放射線の計測では、イベントは等間隔に起こるわけではなく、イベントの起こる間隔にバラツキがある。そこで、読みだし側との速度差を吸収するためにバッファが用いられる。バッファへのデータ転送は SpaceWire よりもはるかに高速に行なえるため、バッファが溢れていない限りはあるイベントが起こってからすぐに次のイベントに対応することができる。そして、イベントの間隔が長い時間にまとめて

データを読み出せば良い。ただし、平均のイベントレートが平均の読み出し速度を下回っていないとバッファが溢れるため、いったんバッファが溢れると、バッファの内容を全て読み出すまでの時間が長いデッドタイムになってしまう。このようなバッファを実現するため、FIFO と呼ばれるメモリと FIFO に書き込むデータを整形する回路、そして FIFO からの読み出しを制御するステートマシンからなる回路ブロックを作成した。

FIFO とは、First-In First-Out のことで、書き込まれたデータを、その順番で出力するといったデータ構造を持つメモリである。ディジタル回路ではしばしば用いられ、SpaceWire のプロトコルチップにも使われている。ここでは、Cyclone 内蔵のメモリを使って FIFO を生成する、開発ツールの機能を利用した。開発ツールでは生成する FIFO の細かい設定が可能であるため、目的に合ったものを容易に生成することができる。また、Cyclone 内蔵メモリはこの他にも様々な用途に使用できる。ここでは、64 bit 幅のデータを 256 個記憶できる FIFO を生成した。図 4.20 に生成した FIFO の動作を示す。FIFO は wrreq 信号が High の間に data にセットされた値をクロック

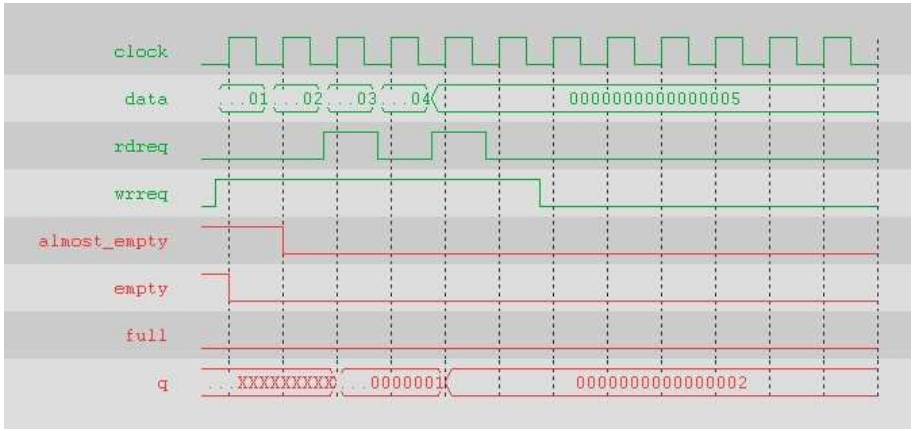


図 4.20: 生成した FIFO の動作

に合わせて読み込む。そして、rdreq 信号が Low から High になる度に、データが読み込まれた順番で queue(q) から出力される。また、FIFO のデータが 1 個以下になると almost empty 信号が Hign になり、FIFO が空になると empty 信号が High になる。そして、FIFO が一杯になると full 信号が High になる。FIFO への書き込みと読み出しは独立しているため、読み出し側は書き込み側の状況を気にする必要が無い。

まず、FIFO に書き込むデータを整形する回路では、測定で得られた 16 bit×4 のデータを単純に連結し、1 つの 64 bit のデータにまとめる。これで FIFO が 1 つで済み、回路が単純になる。そして、FIFO からの読み出しを制御するため、以下のようなステートマシンを作成した。図 4.21 に状態遷移図を示す。また、図 4.22 に、データ読み出し用レジスタのデータフォーマットを示す。

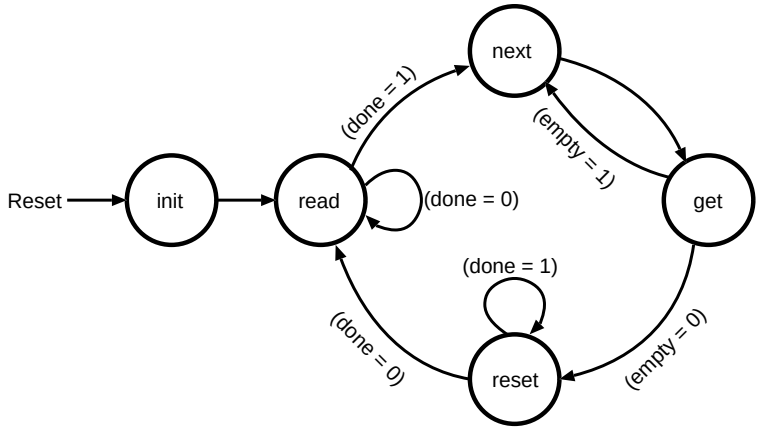


図 4.21: バッファ読み出しの状態遷移図

	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
data_peak1_reg	data	X	OTR	A/D ch1 13 - 0 bit													

	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
data_peak2_reg	data	X	OTR	A/D ch2 13 - 0 bit													

	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
data_time_h_reg	data	trig 2	trig 1	X				Time 23 - 16 bit									

	bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
data_time_l_reg	data	Time 15 - 0 bit															

図 4.22: 読み出しレジスタのデータフォーマット

状態 read

リセットがかかると、まず状態 read になる。状態 read では、SpaceCube が前のデータの読み出しが終わるまで待つため、done 信号が 1 になるまで待機する。done 信号は、SpcaeCube が 1 つ分のデータの取得が完了した際に SpaceCube 側からレジスタに書き込まれる信号である。SpaceCube 側では、ch1 と ch2 の AD 変換値が読み出されるレジスタを常に監視しており、このレジスタの値が 0x0000 以外であれば有効なデータがあるものとみなして読み出し動作を行なう。FADC ボードの ADC は、およそ-2 V から 2 V の範囲を AD 変換するため、何も入力が無いときは 0x2000 付近が出力される。製作した MCA では負の入力は想定していないため、AD 変換した値が 0x2000 を大きく下回る値になることは無く、さらに入力が AD 変換の下限値を越えると out of range(OTR)

信号がADC出力の15 bit目に付加され、ADCの出力値が0x0000になることはない。そのため、done信号が入力されると、次のデータがFIFOから出力されるまでの間有効なデータが無いことをSpaceCubeに伝えるため、AD変換した値が読み出されるレジスタに0x0000を代入して次の状態next に遷移する。

状態 next
状態 next では、FIFOに読み出し要求信号を出力して次の状態 get に遷移する。

状態 get
状態 get では、もしFIFOが空で無ければqueueにデータが出力されるため、出力された64bitのデータを再び16bitずつ4つに分割し、それぞれのデータが入るレジスタに代入し、次の状態reset に遷移する。もしFIFOが空でempty信号が出力された場合には、状態 next に戻り、FIFOにデータが入って来てempty信号が0になるまで読み出しの要求を続ける。

状態 reset
状態 reset では、SpaceCubeが次のデータを読み出す準備ができるまで待つため、done信号が0になるまで待機して、状態 read に遷移する。SpaceCubeではdone信号をレジスタに書き込んだ後、すぐにdone信号を0に戻す。このようにすることで、SpaceCubeが次のデータを読み出す準備ができていることを確認する。

4.3.3 MCAの開発 (ソフトウェア)

FADCボード上に作成した回路をMCAとして動作させるために、SpaceCube側のソフトウェアも作成する必要がある。ソフトウェアはC言語を用いて作成した。また、ソフトウェアの機能のうち、SpaceWireを使ったリモートメモリアクセスについては、能町教授が作成されたライブラリを用いた。

まず、測定を開始する前に時刻カウンタやFIFOをリセットする必要がある。作成した回路にはリセット用のレジスタが用意してあり、このレジスタに値を書き込むことでリセットの情報を伝える。そして、done信号が入力されないとFIFOの読み出しが始まらないので、ここで一回done信号を入力しておく。測定が始まると、FIFOにデータが入ってきて、FIFOの出力がレジスタに代入される。SpaceCubeではこの出力レジスタの値を常に監視して、有効なデータであることを確認したらそのデータを取り込む。その後done信号をレジスタに書き込み、再びdone信号を0に戻すことで1つのデータの読み込みが終わる。

これでMCAの基本的な動作は実現できるのだが、実際に実験室で測定に使用するには、リアルタイムでスペクトルが見えたり、データになんらかの加工を施しながら保存できることが望ましい。しかし、FADCからのデータの読み出しは、SpaceCubeの処理速度に依存しており、SpaceCube側であまり重い処理をさせるとFIFOが溢れる頻度が高くなってしまう。さらに、SpaceCubeのCFの容量は限られており、あまり大量のデータを保存することはできない。そこで、SpaceCube側をサーバとして、TCP/IPプロトコルを用いてネットワーク(100BASE-T)を通して他のPCに

リアルタイムでデータを送り、スペクトルの表示やデータの保存はその PC 上で行なうこととした。このようにすることで、SpaceCube はデータの取り込みと、ネットワークでの通信だけを行えば良く、読み込み速度の低下を防ぐことができる。SpaceCube も相手先の PC も OS として Linux を使っており、このようなネットワークを使ったアプリケーションを容易に実現することができる。

SpaceCube と通信を行ない、データの表示、保存を行なうクライアントプログラムは、ROOT をライブラリとして用いて C++言語で作成した。ROOT を用いることで、ヒストグラムの表示、データの加工などを容易に行なうことができる。また、この後の解析作業でも ROOT を使うため、ROOT のファイル形式でデータを保存できれば便利が良い。なお、クライアント PC の OS には Vine Linux 3.2 を用いた。

以上のようにして、FADC ボードを SpaceWire で制御し、さらにそのデータをネットワーク経由で PC に転送し、遠隔制御で測定できるシステムを構築することができた。SpaceCube を用いれば、SpaceCube 以外に VME BUS に接続する SpaceWire ボードなどの特殊な機器を用意することなく、一般的なネットワーク機器だけで PC にデータを取り込むことができ、非常に安価に SpaceWire を使った計測システムを構築できる。

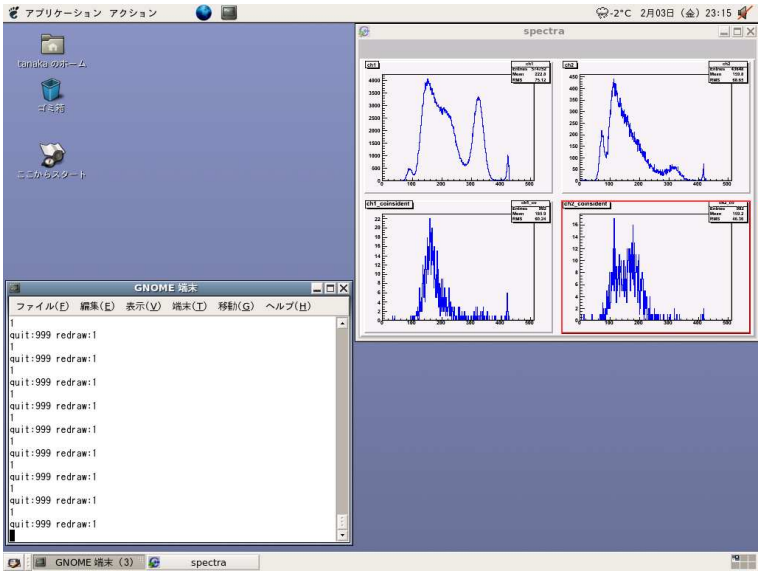


図 4.23: データ収集中の PC 画面

4.4 製作した MCA の評価

製作した MCA が設計通りに動作し、仕様を満たす性能を発揮するかどうかの評価を行った。

4.4.1 リニアリティ

まず、ADC のリニアリティのチェックを行なった。搭載されている ADC は 1 つであるが、マルチプレクサの切替えなどの影響が出ていないかを調べるため、ch1 と ch2 それぞれについてリニ

アリティを調べた。

測定は、Pulser を用いて 300 mV から 1600 mV まで、100 mV 間隔で波高を変化させた 1 kHz の信号を作り、ch1、ch2 についてそれぞれの電圧で 30 秒間ずつデータを取り、各電圧で得られた ADC channel の平均値をその電圧での ADC channel として入力電圧と ADC channel の線形性を調べた。

図 4.24 と図 4.25 にそれぞれのチャンネルでの測定結果を示す。ここで得られている ADC channel は、14 bit の ADC の出力から 0x2000 を引き、それを 2 で割ることで 0 V から 2 V の範囲を 4096 channel で表すようにしたものである。測定結果から、ch1、ch2 ともリニアリティには特に問題が無いことが分かった。

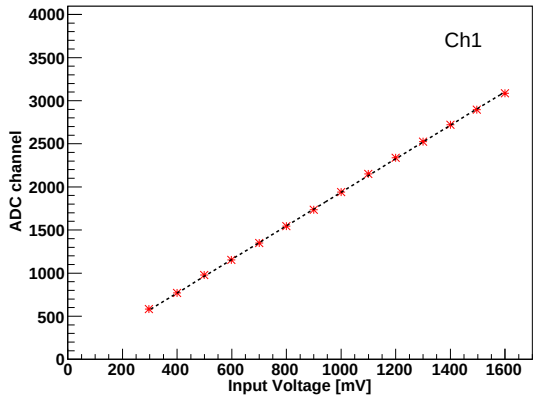


図 4.24: ch1 のリニアリティ

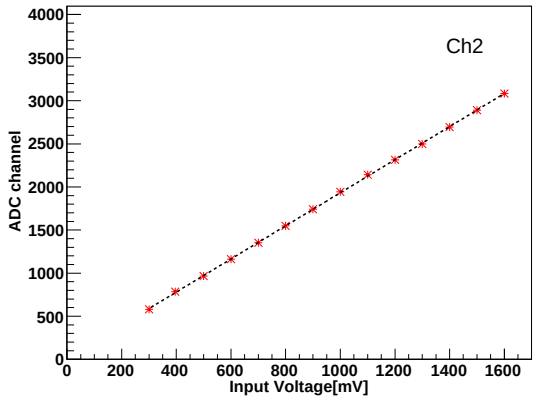


図 4.25: ch2 のリニアリティ

4.4.2 デッドタイム

バッファが溢れていない場合の 1 イベントごとのデッドタイムを、オシロスコープで busy 信号を観測することによって測定した。図 4.26 に示すように、1 イベントごとのデッドタイムは 13.6 μs となった。これは VHDL ソースから計算される値とほぼ一致する。これにより、busy 信号が正しく出力されていることが確認できた。また、要求仕様である 10 kHz のイベントレート時にバッファが溢れていなければ、デッドタイムの割合が 13.6%となる。一般に MCA ではデッドタイムの割合が 30%から 40%程度までならば許容できるとされているため、それと比較すれば十分に短いデッドタイムである。ただし、ここで測定されるデッドタイムは FADC ボードのみのデッドタイムであり、実際の測定ではアナログ回路で生じるデッドタイムも存在することから、ここで測定されたデッドタイムが必ずしもアナログ回路まで含めた測定系全体でのデッドタイムに一致するとは限らないことに注意する必要がある。

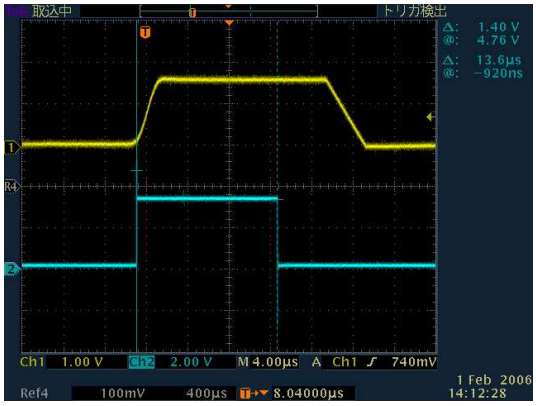


図 4.26: 1 イベントごとのデッドタイム
上がピークホールド出力、下が busy 信号

次に、時刻カウンタとデッドタイムカウンタの動作を調べた。測定は Pulser により生成した 200 Hz の信号を MCA の一方のチャンネルに入力し、測定後の全イベント数とデッドタイムカウンタの値を比較することで行なった。この程度のイベントレートではバッファが溢れることはなく、イベント数に 1 イベントあたりのデッドタイムをかけたものが測定全体のデッドタイムと等しくなる。測定で得られた値は表 4.2 のようになった。全イベント数と 1 イベントごとのデッドタイムから、この測定でのデッドタイムは 1.67 s である。一方デッドタイムカウンタの値とデッドタイムカウンタのクロックが 1.50 MHz であることから、計算されるデッドタイムの値は 1.72 s である。これは予想された値よりもやや大きい値となっているが、デッドタイムの測定の最小単位が 677 ns であるため、13.6 μs のデッドタイムが 14.2 μs として記録されているためであり、設計通りの動作である。これよりデッドタイムカウンタは正しく動作しており、実際の値と誤差は生じるものの十分な精度でデッドタイムを計測できることが分かった。また、時刻カウンタのクロックが 1464.84375 Hz であることから、このときの計測時間は 613.5 s と計算される。全イベント数をこの時間で割ると 200 Hz となり、設定したイベントレートに等しくなる。よって時刻カウンタも正しく動作していることが確かめられた。

表 4.2: 時刻カウンタとデッドタイムカウンタのテスト

全イベント数	122704
時刻カウンタの進み	898687
デッドタイムカウンタの進み	2576805

4.4.3　トリガの判定

作成した MCA では、一方のチャンネルでトリガが立ってから 6 μs 以内にトリガが立った場合に同時にトリガが立ったものと判断されるはずである。そこで、これを確かめる実験を行った。実験は 2 台の Pulser を用意し、片方の Pulser のトリガ出力ともう一方の Pulser のトリガ入力の間に Delay Generator を入れ、任意の間隔でイベントを発生させることができるようにしておき、MCA の出力結果をモニタしながらイベントの間隔をずらしてゆくことで行った。また、イベントの間隔はオシロスコープで測定した。その結果、6 μs を境にして MCA の出力結果ではトリガが同時と判断され、それ以上間隔が空くと ch1 と ch2 のイベントが交互に表れるようになったことから、トリガの判定は正しく行われていることが確認できた。

4.4.4　スピード試験

ここでは、バッファが溢れないイベントレートの限界、イベントの取りこぼしが無いかの確認、イベントレートが高くなったときのスペクトルの変化について調べ、イベントレートが高くなったときに作成した MCA が正しく動作するのかを調べた。

まず、バッファが溢れる限界のイベントレートの目安を調べるため、Pulser からの信号を ch1 のみに入力し、busy 信号をオシロスコープで監視しながら Pulser のレートを上げてゆき、busy 信号に異変を生じるイベントレートを調べた。その結果、およそ 14 kHz までバッファが溢れることなく測定できた。図 4.27 にそのときの busy 信号とピークホールド信号の様子を示す。ここでさらにイベントレートを上げてゆくと、図 4.28 に示すようにバッファが溢れ、バッファの中のデータが 1 個になるまで書き込みを中断するため、busy 信号が長い時間立ったままになる。イベント

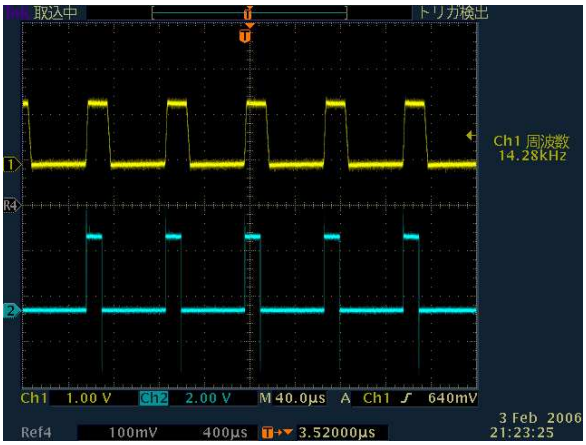


図 4.27: バッファが溢れる直前の busy 信号

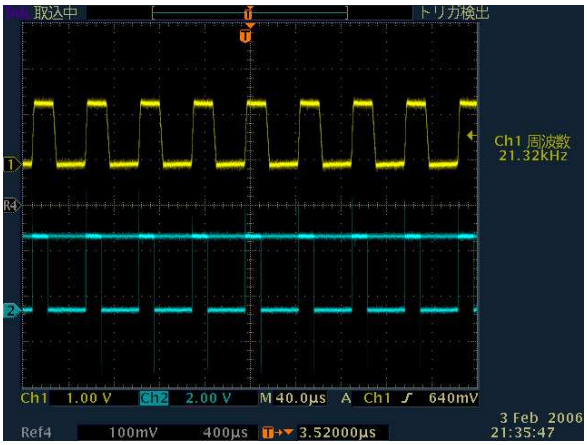


図 4.28: バッファが溢れたときの busy 信号

レートが 14 kHz のときのデッドタイムの割合は 19%となるため、平均のイベントレートが 2ch 合わせて 14 kHz 程度でも問題なく測定を行うことができるものと考えられる。また、SpaceWire を通した読み出し側も 14kHz 程度の速度で読み出せていることになるため、このときのスループットは、一度のイベントで読み出すデータが 64bit であることから、896 kbps となる。SpaceWire

の仕様上はさらに高速に通信できるはずであるが、この場合は SpaceWire の通信速度ではなく、SpaceCube での処理速度によって読み出しの速度が決まっているものと思われる。そのため、読み出しプログラムを効率化することで、読み出しをさらに高速化できる可能性が高い。

次に、イベントのトリガを Visual Scaler によって数え、MCA で測定されたイベント数と比較することで、イベントの取りこぼしが発生していないかを調べた。測定は Pulser で 12 kHz の信号を作り、ch1 のみ、ch2 のみ、ch1 と ch2 両方について行った。ch1 と ch2 両方に信号を入力する際は、2 台の Pulser を同期させて信号を入力した。結果は表 4.3 のようになった。Scaler の測定開始や終了の操作を手動で行ったことと、PC のキーボード操作をしてから実際に測定が開始されたり終了するまでにタイムラグがあることから、Scaler の値と MCA で測定されたイベント数は完全には一致しないが、全ての場合においてほぼ一致しており、イベントの取りこぼしは発生していないものと思われる。

表 4.3: Scaler とのイベント数比較

	Scaler	MCA	ずれ (%)
ch1	10251127	10248060	0.030
ch2	-	-	-
ch1	-	-	-
ch2	10254520	10254720	0.002
ch1	10991947	10989890	0.019
ch2	10991947	10989890	0.019

2 つのチャンネルに同期させた信号を入力した場合、バッファが溢れたり 1 イベントごとのデッドタイムより短い周期の信号が入ったりしない限り、デッドタイム中にイベントが発生することはない。そこで、二つのチャンネルに同期していない信号を入力した場合について調べた。この場合はデッドタイム中にイベントが発生する場合があるため、Scaler の値より少ないイベント数が測定されるはずである。測定は、ch1 に 7.0 kHz を、ch2 に 5.0 kHz の同期していない信号を入力して行った。測定結果を表 4.4 に示す。予想通り、測定された値は Scaler の値より明らかに小さい。また、ch1 の測定に影響するデッドタイムは ch2 のイベントで発生したものであり、ch2 の測定に影響するデッドタイムは ch1 のイベントで発生したものであるため、ch1 より ch2 の方がずれが大きくなっている。

表 4.4: Scaler とのイベント数比較 (ch1 と ch2 が非同期)

	Scaler	MCA	ずれ (%)
ch1	10112941	9730769	3.78
ch2	7171828	6790419	5.32

次に、イベントレートが高くなった場合にスペクトルに影響が出ないかどうかを調べた。まず、10 kHz と 30 kHz の信号を Pulser で作り、ch1 にのみ入力した場合について調べた。その結果、図 4.29 と図 4.30 に示すように、イベントレートが 10 kHz と 30 kHz の間に大きな違いは見られなかった。よって、この程度のイベントレートではスペクトルの形に影響は無いと言える。

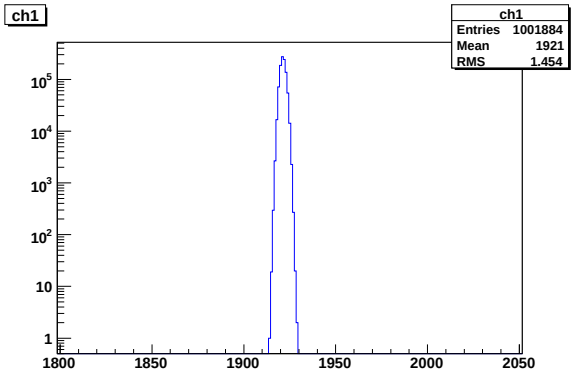


図 4.29: イベントレート 10 kHz でのスペクトル

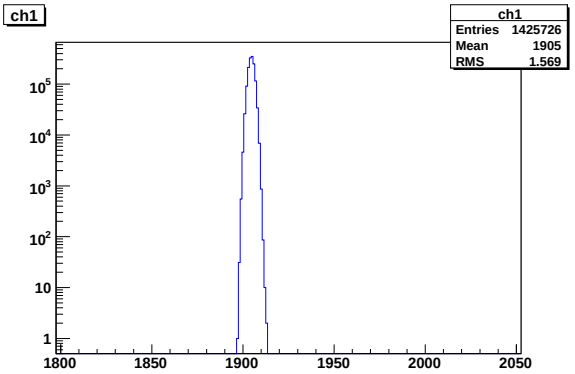


図 4.30: イベントレート 30 kHz でのスペクトル

次に ch1 と ch2 に同期していない信号を入れた場合のスペクトルを調べた。まず、ch1 に 2.0 kHz の信号を、ch2 に 1.0 kHz の信号を入れた場合のスペクトルを図 4.31 と図 4.32 に示す。スペクトルに乱れが見られるが、これは FADC ボード側の問題ではなく、ピークホールドモジュールにおいて、あるイベントでピークホールドしている途中で新たなイベントが入ったために前のイベントのピークホールドが途中で打ち切られ、ピークホールド出力が乱れる現象が発生したために生じたものである。また、図 4.33 と図 4.34 に、ch1 に 7.0 kHz の信号を、ch2 に 17 kHz の信号を入れた場合のスペクトルを示す。イベントレートが上がったことにより、ピークホールド出力が乱れる頻度が高くなっている。

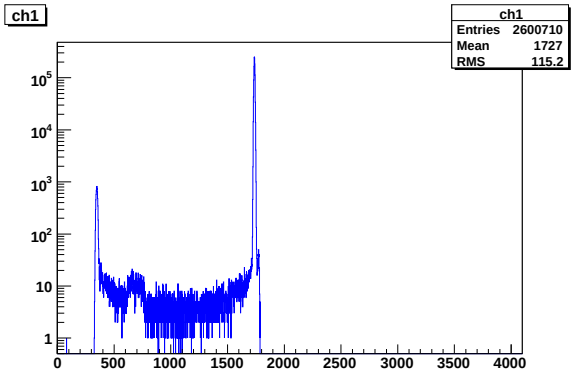


図 4.31: 同期していない信号 (2.0 kHz) を入れたときの ch1 のスペクトル

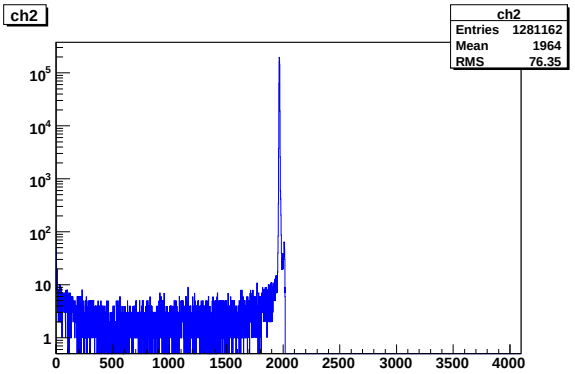


図 4.32: 同期していない信号 (1.0 kHz) を入れたときの ch2 のスペクトル

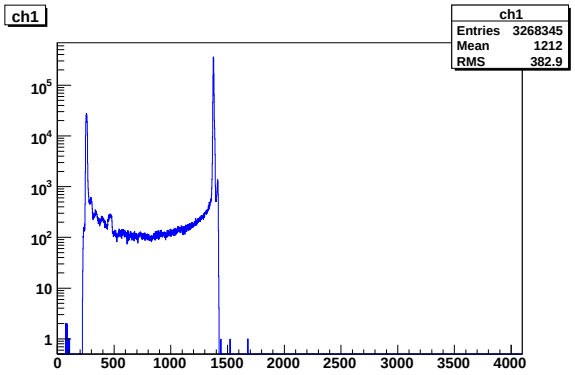


図 4.33: 同期していない信号 (7.0 kHz) を入れたときの ch1 のスペクトル

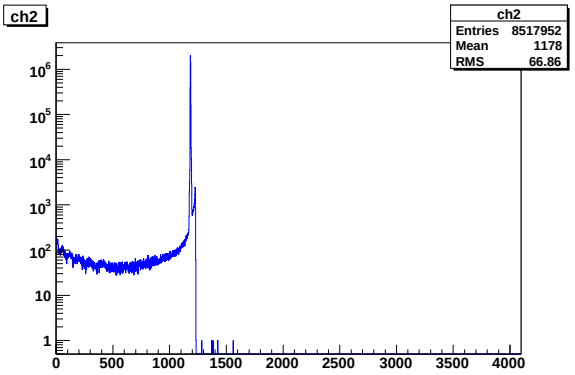


図 4.34: 同期していない信号 (17 kHz) を入れたときの ch2 のスペクトル

ここまでの試験で、製作した FPGA の回路は設計通りの動作をしており、要求された仕様を十分に満たす性能であることが確認された。また、ピークホールドモジュールの動作にやや問題が見られるが、数 kHz 程度のイベントレートまでならばあまり大きな影響はない。

4.4.5 ガンマ線源の測定

製作した MCA の性能が確かめられたことから、実際にシンチレータとフォトダイオードを使った検出器でガンマ線源の測定を行った。測定に使用した検出器は、CsI シンチレータとフォトダイオード、さらにプリアンプまでが一体化され、コンパクトな金属ケースに納められたもので、この検出器出力を Shaper に接続するだけで測定を行うことができる。図 4.35 に検出器の写真を示す。

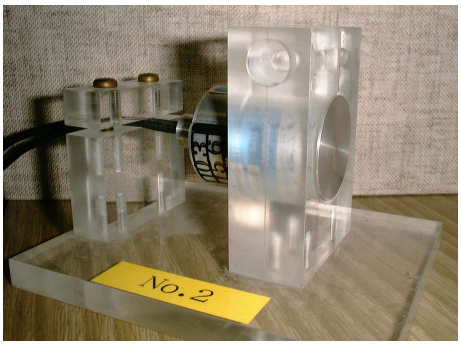


図 4.35: CsI 検出器

スペクトルの取得

まず検出器を 2 台用意し、ch1、ch2 それぞれの Shaper に接続した後、Shaper のゲイン、ピークホールドモジュールの threshold を調整した。そして、^{22}Na と ^{137}Cs のスペクトルを取得した。測定は 2 つの検出器を向かい合わせに置き、2 つの検出器の中間に線源を置くことで、2 チャンネル同時に行った。

図 4.36 と図 4.37 に測定で得られたスペクトルを示す。ここで、スペクトルの右端に小さなピークのようなものが見えるが、これはピークホールドモジュールの最大出力電圧が ADC のフルスケール電圧に満たないために、ピークホールド出力がサチュレーションしている影響が表れているものである。^{22}Na の下側 2 つのスペクトルは、ch1 と ch2 で同時にトリガが立ったときの (coincidence をとった) スペクトルである。^{22}Na は β^+ 崩壊をするため、陽電子が放出される。放出された陽電子はすぐに運動エネルギーを失い、線源のケースなどの物質中の電子と消滅過程を起こす。その結果、電子の静止質量に等しい 511 keV のエネルギーを持った消滅放射線と呼ばれる光子が通常 2 個飛び出すが、この光子は元々の電子と陽電子が持っていた運動量を保存するため、正反対の方向に飛び出す。そのため、^{22}Na のスペクトルを取得する際に向かい合わせに置いた検出器で coinicidence をとると、511 keV の消滅放射線によるスペクトルが現れる。また、^{22}Na が陽電子を放出する過程では 1275 keV の準位に遷移し、511 keV の消滅放射線と同時に 1275 keV のガンマ線も放出する。ただし、このガンマ線光子は単独であらゆる方向に飛び出すため、消滅放射線で coincidence を取ったイベントで現れる確立は非常に低い。そのため、図 4.37 の下側のスペクトルには 1275 keV のピークは現れず、511 keV のピークとそのコンプトン散乱のみが現れている。

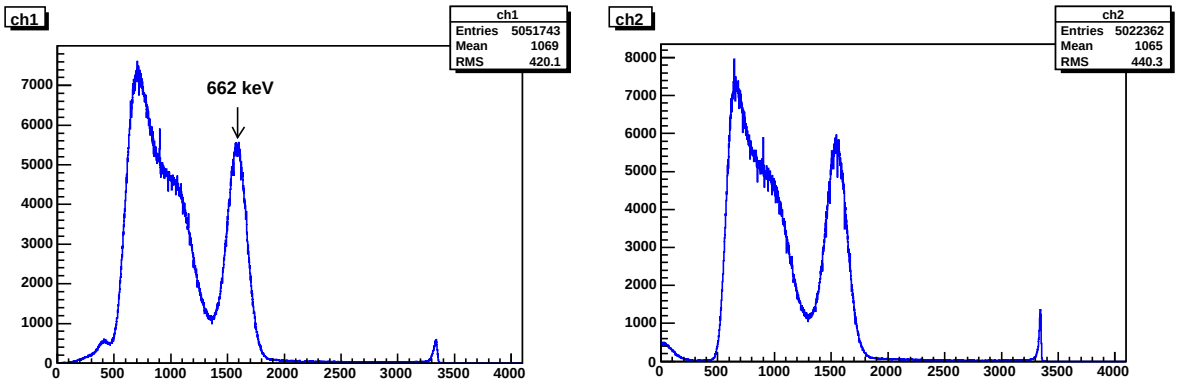


図 4.36: ^{137}Cs のスペクトル

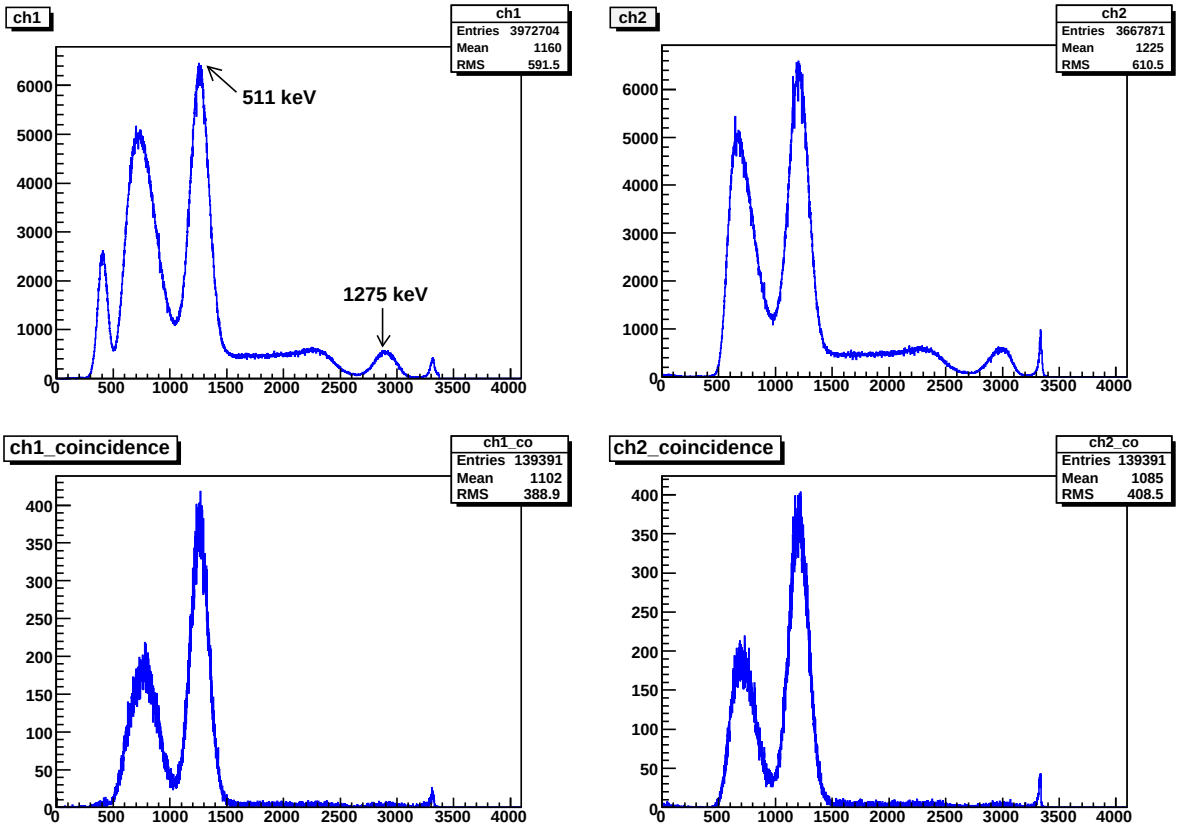


図 4.37: ^{22}Na のスペクトル (下は coincidence を取ったもの)

ここで得られたスペクトルの各ピークをガウシアン関数でフィッティングし、エネルギーと ADC channel を対応させる関数を求めておき、この後のコンプトン散乱の測定で用いた。

コンプトン散乱の測定

次にコンプトン散乱の測定を試みた。測定は、2 つの検出器のうち一方の検出器のシンチレータを散乱体として、もう一方の検出器で散乱光子を検出するという方法で行った。図 4.38 に測定のセットアップを示す。検出器 1 は MCA の ch1 に、検出器 2 は MCA の ch2 に接続されている。ガンマ線源には ^{137}Cs の 662 keV を用い、線源は鉛容器に入れ、散乱光子を検出する検出器には線源のガンマ線が直接入らないようにした。コンプトン散乱イベントを検出すると、両方の検出器で同時にトリガが立つはずであるから、他のイベントやバックグラウンドからコンプトン散乱イベントだけを分離することができる。

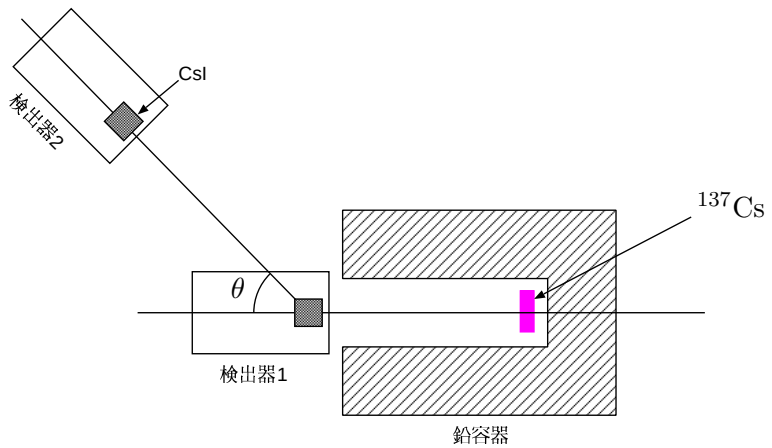


図 4.38: コンプトン散乱測定のセットアップ

元のガンマ線光子のエネルギーを E、散乱光子のエネルギーを E_2 とすると、E と E_2 の間には、

$$E_2 = \frac{E}{1 + \frac{E}{m_0 c^2}(1 - \cos\theta)} \tag{4.1}$$

の関係が成り立ち、散乱体で落とされるエネルギー E_1 は、$E_1 = E - E_2$ となる。ここで、m_0c^2 は電子の静止エネルギーで、511 keV である。θ は 90 度と 45 度について測定を行ったので、測定される散乱光子のエネルギーは 90 度のとき 288 keV、45 度のとき 480 keV となるはずである。そして、散乱体の検出器で落とされるエネルギーは、それぞれ 374 keV、182 keV となるはずである。

図 4.39 に θ が 90 度のときのスペクトルを、図 4.40 に θ が 45 度のときのスペクトルを示す。ここで、各図のうち左上が散乱体の検出器で得られたスペクトル、右上が散乱光子を受ける検出器で得られたスペクトルである。そして、2 つの検出器で同時にトリガが立ったイベントだけのスペクトルが左下 (散乱体) と右下 (散乱光子) である。coincidence を取ったスペクトル中には、あらかじめ予想したエネルギーに対応した ADC channel が点線で示されている。

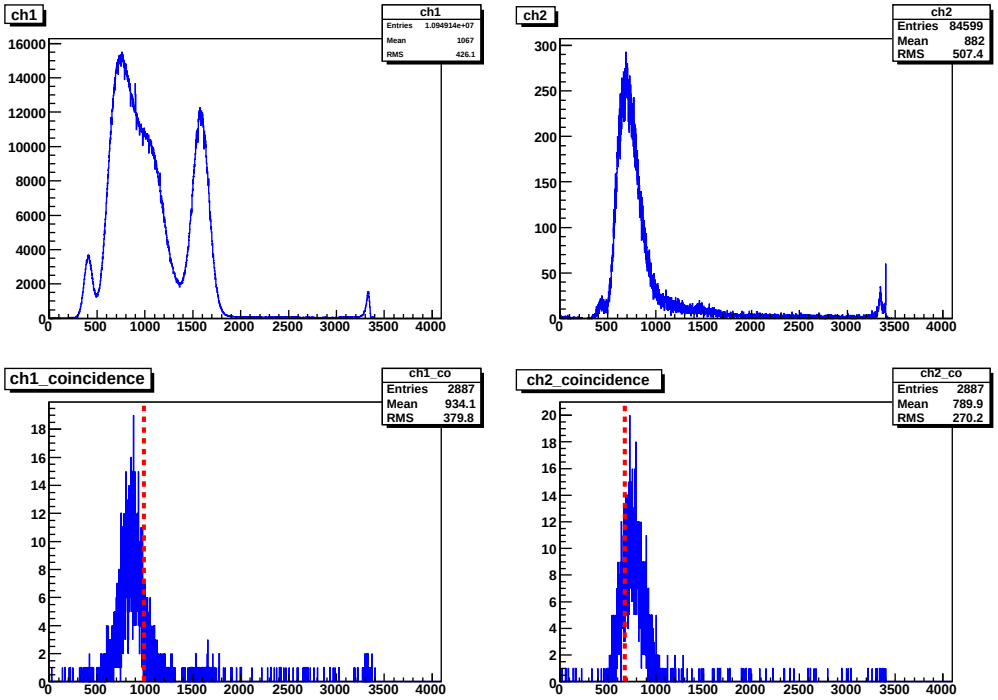


図 4.39: θ が 90 度のときのスペクトル

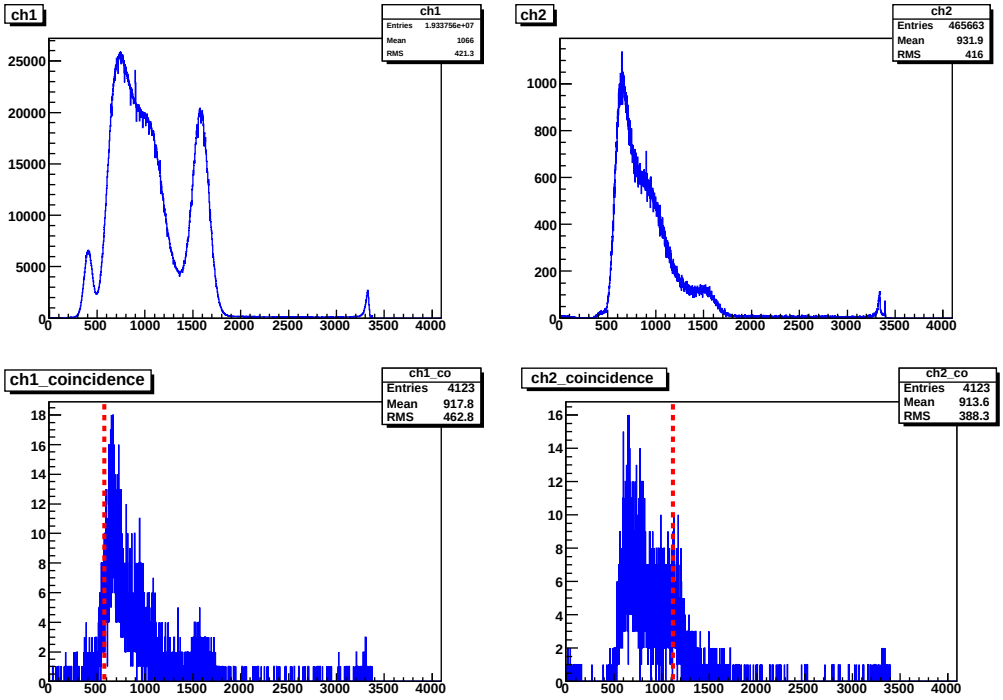


図 4.40: θ が 45 度のときのスペクトル

coincidence を取ったスペクトルを見ると、予想されたエネルギー付近にピークらしきものが見られる。しかし、たまたま同時に起こったと思われるバックグラウンドのイベントもかなり多く、コンプトン散乱によるピークがあまりはっきりしていない。そこで、同時に起こったイベントについて、それぞれの検出器で測定されたエネルギーを用いて 2 次元ヒストグラムを作成した。コンプトンイベントならば 2 つの検出器で測定されるエネルギーの間には $E = E_1 + E_2$ の関係が成り立っており、他のイベントと明確に区別できるはずである。図 4.41 と図 4.42 に作成したヒストグラムを示す。コンプトンイベントならば図中の点線で表された直線に乗っているはずであるが、θ が 90 度のときは明確にコンプトンイベントを分離できており、θ が 45 度のときは、ややバックグラウンドが多いながらも、円で示したように分離することができている。この結果から、確かにコンプトン散乱イベントを測定できていることが分かった。

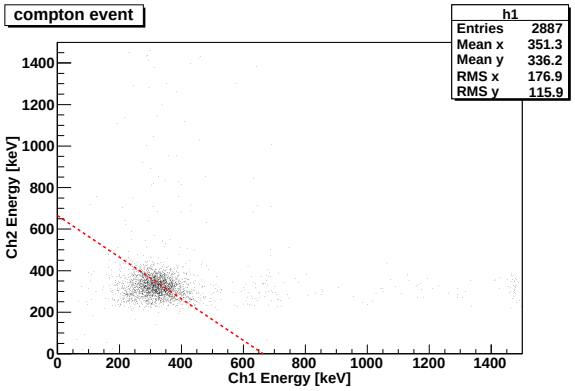


図 4.41: θ が 90 度のときの 2 次元ヒストグラム

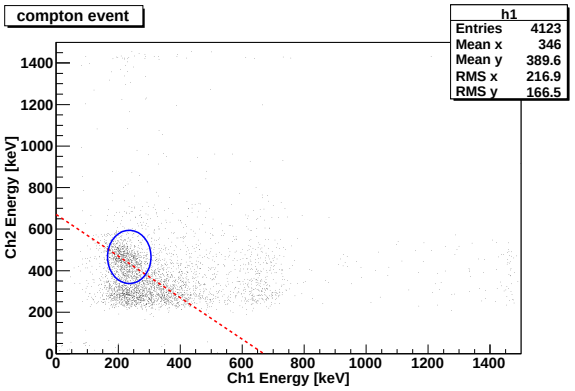


図 4.42: θ が 45 度のときの 2 次元ヒストグラム

以上のように、開発した MCA を用いてガンマ線のスペクトルを取得したり、トリガパターンを用いて coincidence を取りコンプトン散乱を測定することができた。この結果から、SpaceWire を使った MCA としてかなり実用的なものを開発することができたと言える。

第5章　まとめと今後

本研究では、半導体検出器の原理に始まり、DSSD、コンプトンカメラ、DSSD の多チャンネル読み出しシステム、SpaceWire について理解を深め、SpaceWire を使った装置として、マルチチャンネルアナライザを開発した。

DSSD の多チャンネル読み出しシステムについては、現在使用されている読み出しシステムを用いて、実際に 4 cm DSSD の立ち上げから測定、解析までを行った。その結果、4 cm DSSD の評価を行える状態にすることができたが、現在の読み出しシステムが抱える問題点の数々が明らかになった。そして、今後導入される予定である新しい読み出しシステムについて調べ、その読み出しシステムが現在使われている読み出しシステムの問題点をうまく改善したものであることが分かり、導入後の立ち上げがスムーズに行えるよう、知識面での準備ができた。

SpaceWire については、簡単なハードウェア構成であるにもかかわらず、高い柔軟性と信頼性を備えた規格であることが分かった。そして、SpaceWire と FPGA、VHDL の理解を深めるために汎用的な ADC ボードを用いて MCA を製作したが、完成した MCA はかなり高性能なもので、十分実用に耐えるものとなった。SpaceWire を積極的に利用した測定装置がまだあまり普及していない中で、早い段階から SpaceWire の取扱いを経験し、測定装置に応用できたことは、今後 SpaceWire を搭載した DSSD 他チャンネル読み出しシステムの開発を進める上で有利な点である。

今後については、今回得られた DSSD や多チャンネル読み出しシステムの知識を生かして、間もなく導入される予定である DSSD の新しい読み出しシステムの立ち上げにあたりたい。また、MCA の開発中に思い付いたアイディアで、本研究に盛りこめなかったものがいくつかあるため、それらの具体化を試みたい。そして、今後も SpaceWire や VHDL、FPGA についての理解を深め、DSSD の多チャンネル読み出しシステムの SpaceWire による制御を早期に実現したい。

謝辞

本研究を進めるにあたり、大杉教授には、DSSD の基礎から研究室での生活全般まで、大変お世話になりました。深く感謝致します。また、深沢助教授には研究の進め方から論文の書き方まで、細かいところまで丁寧に指導して頂きました。さらに研究のための環境整備にも尽力して頂き、深く感謝致します。JAXA/ISAS の高橋教授には、SpaceWire について学ぶ機会を与えてくださり、研究に関する助言を頂きました。深く感謝致します。また、高橋研の渡辺さんには SpaceCube や FADC ボードの立ち上げについてお世話になりました。そして、同じ高橋研の武田さんには、4 cm DSSD について助言を頂きました。深く感謝致します。大阪大学の能町教授には、SpaceWire や FADC ボードに関する質問に丁寧に答えて頂きました。さらに、能町教授の作成された VHDL ソースから学んだことは非常に多く、深く感謝致します。

M2 の澤本さん、M1 の安田さんには、DSSD の基礎から読み出し、解析まで、たくさんのことを教えて頂きました。深く感謝致します。その他研究室のみなさんには、何かとお世話になり、有意義な時間を過ごすことができました。ありがとうございました。また、事務室の石井さん、上原さんには、各種事務手続きに関して大変お世話になりました。深く感謝致します。

関連図書

[1] ALTERA, Cyclone Device Handbook, Volume 1

[2] Analog Devices, AD9240 Data sheet, 1998

[3] Analog Devices, ADG608/ADG609 Data sheet, 1995

[4] Bjorn Magne Sundal, The VA-DAQ 1.2 test and readout system for the VA/TA chip sets, 1999

[5] ECSS, SpaceWire Draft ECSS-E-50-12A, 2002

[6] Glenn K.Knoll 木村逸朗 阪井英次 訳 「放射線計測ハンドブック 第 2 版」, 日刊工業新聞社,1991

[7] ideas, VA-DAQ VA-TA Evaluation

[8] ideas, VA32TA Version 0.92

[9] Texas Instruments, SN74LVT244B Data sheet, 2003

[10] 久保信「FEC-IFC バス規格案 Ver. 1.0」, 2004

[11] 澤本直之 「シリコンストリップ検出器の X 線応答と多チャンネル読み出し回路のパラメータ調整」, 広島大学 卒業論文, 2004

[12] 次期 X 線天文衛星計画ワーキンググループ「NeXT 計画提案書」, 2005

[13] 高橋忠幸 笠羽康正 高島健 吉光徹雄 山田隆広 能町正治 「科学衛星データ処理系の将来展望」

[14] 中本達也 「多層シリコンストリップと BGO・フォトダイオードを用いた宇宙軟ガンマ線コンプトンカメラの開発」, 広島大学 学位修士論文, 2004

[15] 長谷川裕恭 「改訂・VHDL によるハードウェア設計入門」, CQ 出版社, 2004

[16] 森岡澄夫「HDL による高性能ディジタル回路設計」, CQ 出版社, 2002